Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

**MASTER EN SCIENCES INFORMATIQUES**

# Amélioration de la qualité des processus logiciels dans les PME expérimentation, amélioration et outillage

Grégory, Cassiers

*Award date:*
2007

*Awarding institution:*
Universite de Namur

Link to publication

**General rights**
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

**Take down policy**
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

**Facultés Universitaires Notre-Dame de la Paix**
**Institut d'Informatique**

# Amélioration de la Qualité des Processus Logiciels dans les PME

*Expérimentation, Amélioration et Outillage*

**Cassiers Grégory**

*Mémoire présenté en vue de l'obtention du grade de maître en informatique*


**Année Académique 2006-2007**

**Institut d'Informatique - 21 rue Grandgagnage 5000 Namur**
**Tél. : 081/72.41.11 - Fax : 081/72.49.67**

# Résumé

Les modèles visant l'amélioration des processus et pratiques logiciels, tels que CMMI ou SPICE, ne sont pas adaptés aux petites et très petites entreprises de par leur lourdeur et leurs coûts. C'est pourquoi, les Facultés Universitaires Notre-Dame de la Paix de Namur ont développé une approche visant ce type d'entreprise : OWPL[1]. Ce mémoire présente cette approche ainsi qu'une des méthodes le composant, à savoir la Micro-Évaluation, via différentes démarches d'amélioration entreprises dans des petites entreprises québécoises. Ensuite, ce mémoire reprend les différentes améliorations devant être apportées à cette méthode ainsi que l'élaboration d'un outil statistique pour sur base de celle-ci.

# Abstract

Models to improve software processes and practices, such as CMMI or SPICE, are not adapted to small and very small companies because of their heaviness and costs. This is why the Faculty of Namur developped an approach for these companies : OWPL[2]. This thesis presents this approach and one of its methods. In others words, it explains the Micro-Evaluation and exemplifies it through several improvement steps in small companies from Quebec. Moreover, this thesis highlights improvements that should yet be bought to the method aswell as the development of a statistical tool based on the Micro-Evaluation.

[1] **O**bservatoire **W**allon des **P**ratiques **L**ogicielles
[2] **O**bservatoire **W**allon des **P**ratiques **L**ogicielles

# Avant-propos

Ce document est l'aboutissement d'un stage de fin d'études à l'École de Technologie Supérieure (ETS) de Montréal, Québec, Canada. Il est présenté en vue de l'obtention du grade de maître en informatique aux Facultés Universitaires Notre-Dame de la Paix par un étudiant de dernière année de maîtrise. Les résultats présentés n'ont pu être obtenus que grâce à un accord bilatéral entre ces deux établissements.

La rédaction d'un mémoire ne pouvant se réaliser seul, nous tenons tout d'abord à remercier l'ensemble des personnes qui ont participé de près ou de loin à cette rédaction.

Nous tenons tout d'abord à remercier le Professeur Naji Habra, promoteur de ce mémoire en Génie Logiciel, pour son suivi durant tout notre stage ainsi que pour son aide à la rédaction de ce mémoire.

De même, nous remercions le Professeur Jean-Marc Desharnais de l'École de Technologie Supérieure, maître de notre stage, et Mohammad Zarour, étudiant en doctorat à l'École de Technologie Supérieure, pour l'encadrement et le soutien qu'ils nous ont fournis durant tout notre séjour au Canada.

Nous remercions bien entendu les dirigeants des deux entreprises québécoises, à savoir Club Capra et Horizon Vert Centre-du-Québec, pour le temps qu'ils ont bien voulu nous consacrer afin que nous puissions réaliser notre stage.

Nous tiendrons aussi à remercier les différentes personnes ayant consacrées une partie de leur temps à le relecture de ce mémoire, à savoir Alain Mordant, François Clément et Michelle Wallez.

Nous terminerons par remercier tous nos proches pour leur soutien et leur aide tout au long de cette dernière année d'étude.

# Table des matières

# Table des figures

# Abréviations et sigles

Sont repris dans la liste ci-dessous les abréviations et sigles les plus utilisés dans ce mémoire.

**CETIC** **C**entre d'**E**xcellence en **T**echnologies de l'**I**nformation et de la **C**ommunication, Université de Namur, Belgique. http ://www.cetic.be

**CMM** **C**apability **M**aturity **M**odel, Software Engineering Institute (SEI), Carnegie Mellon University, http ://www.sei.cmu.edu/cmm

**CMMI** **C**apability **M**aturity **M**odel **I**ntegration, Software Engineering Institute (SEI), Carnegie Mellon University, http ://www.sei.cmu.edu/cmmi

**ISO** **I**nternational **S**tandards **O**rganization, http ://www.iso.org

**OWPL** **O**bservatoire **W**allon des **P**ratiques **L**ogicielles du Centre d'Excellence en Technologies de l'Information et de la Communication (CETIC), Université de Namur, Belgique. http ://www.info.fundp.ac.be/~software-quality/fr/owpl/demarche.htm

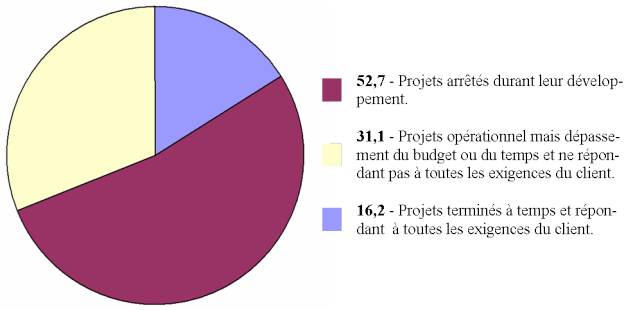
**SEI** **S**oftware **E**ngineering **I**nstitute, Carnegie Mellon University, http ://www.sei.cmu.edu

**SPICE** **S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination, International Standards Organization (ISO), http ://www.sqi.gu.edu.au/spice/

# Introduction

Le secteur du logiciel est constitué principalement de petites et très petites entreprises. En effet, comme le rapporte Laporte et al [15], 85% des sociétés du secteur logiciel européen sont composées de 1 à 10 employés. Cette constatation ne s'applique pas uniquement à l'Europe puisqu'au Canada, dans la région montréalaise, 80% des organisations oeuvrant dans le domaine du logiciel ont moins de 25 employés. De même qu'au Brésil où 70% des entreprises informatiques sont des petites ou très petites entreprises [16].

Cette composition du marché informatique implique une forte problématique : le manque de qualité des logiciels produits ou des services offerts par une entreprise. En effet, selon [25], plus de 80% des projets informatiques ne sont pas achevés dans le laps de temps demandé ou ne possèdent pas toutes les fonctionnalités requises par le client, comme le montre la figure ci-dessous tirée de [25]. Ces échecs sont dus à une mauvaise gestion de l'entreprise au point de vue de son management, de sa relation avec ses clients ou encore de sa méthodologie de développement.




Depuis maintenant plusieurs années, des modèles, normes et référentiels ont été développés avec, pour objectif, l'amélioration de la qualité des processus et pratiques d'une entreprise oeuvrant dans le secteur logiciel. Le problème est que ces outils tels que Capability Maturity Model (CMM) ou Software Process Improvement and Capability dEtermination (SPICE) ne sont qu'adaptés aux grandes et moyennes entreprises. Une utilisation totale ou partielle de ceux-ci est infaisable pour des petites ou très petites entreprises, de même qu'une éventuelle adaptation.

C'est pourquoi, les Facultés Universitaires Notre-Dame de la Paix de Namur ont développé une approche d'amélioration de la qualité des processus et pratiques logicielles pour les petites et très petites entreprises : OWPL (**bservatoire Wallon des Pratiques Logicielles**). De plus, une méthode a été élaborée sur les mêmes bases : la micro-évaluation [5]. Cette méthode a déjà été appliquée sur plusieurs entreprises wallonnes (Belgique), québécoises (Canada) et françaises [7]. Ces différentes applications ont permis de compléter et affiner la méthode afin qu'elle soit mieux adaptée aux entreprises qu'elle cible.

Ce mémoire a pour objectif de continuer sur cette même voie. C'est pourquoi, deux entreprises montréalaises ont été sollicitées afin d'acquérrir une certaine expérience dans l'application de la micro-évaluation. Nous avons donc pu déclencher un processus d'amélioration pour ces deux entreprises et ainsi, tenter d'améliorer la qualité de leurs processus et pratiques logicielles déjà en place. Via cette expérience, nous avons ensuite été en mesure de critiquer la méthode actuelle et de proposer une nouvelle version de celle-ci. Ce mémoire décrit les différentes étapes par lesquelles nous sommes passés pour arriver à cette nouvelle version.

De plus, un outil statistique a été entrepris dans le but de traiter et utiliser les résultats obtenus par les diverses applications de la micro-évaluation, et ce, dans différents pays. Cet outil, appelé MicroStat, a été développé en collaboration avec l'Institut d'Informatique de Namur et le CETIC (textbfCentre d'**E**xcellence en **T**echnologies de l'**I**nformation et de la **C**ommunication).

La première partie de ce mémoire passe en revue les différents modèles, normes et référentiels permettant d'améliorer la qualité des processus et pratiques logiciels. La deuxième partie décrit notre intervention au sein d'un club d'étudiants de l'École de Technologie de Montréal, Capra. La troisième partie décrit, quant à elle, notre intervention au sein de l'entreprise québécoise Horizon Vert. Les deux dernières parties représentent la partie recherche de notre mémoire, à savoir l'amélioration du modèle québécois de la micro-évaluation et la présentation de l'outil statistique MicroStat.

# Première partie

# Modèles & Méthodes d'amélioration des processus et pratiques logiciels

# Section 1 :

## *Modèles, normes et référentiels*

> **Contenu :** *Présentation des différents modèles, normes et référentiels permettant d'améliorer la qualité des processus et pratiques logiciels d'une entreprise.*

### 1.1 Introduction

Le secteur du logiciel devenant de plus en plus compétitif, les entreprises informatiques ont besoin de se différencier les unes des autres afin de devenir plus performantes. Ces différenciations peuvent porter sur différents aspects tels que la réduction des coûts de production ou de développement ou l'utilisation de stratégies de qualités. C'est pourquoi, ces entreprises tentent d'améliorer leurs pratiques et processus logiciels étant déjà en place. Ces améliorations permettent d'accroître leurs performances et la qualité des services et produits qu'ils proposent.

Dans l'optique d'entreprendre ces démarches d'une manière efficace, des modèles, normes et référentiels ont été établis par différents centres et institutions agréés tels que ISO (International Standards Organization) ou SEI (Software Engineering Institute). Ce sont ces outils qui vont maintenant être présentés.

### 1.2 Principales approches actuelles

Le domaine du logiciel comporte un ensemble de modèles, normes et guides référentiels dont l'efficacité n'est plus à démontrer. Selon [9], les deux approches les plus connues à l'heure actuelle sont CMM (Capability Maturity Model) et SPICE (Software Process Improvement and Capability Determination)[1]. Ces modèles sont en continuelle évolution. Commençons par les décrire brièvement :

*CMM*

Selon [21], ce modèle a été présenté en 1991 par le SEI (Software Engineering Institute) suite à une demande du Département de la Défense des États-Unis afin de pouvoir évaluer ses différents fournisseurs de logiciels. A la base, ce modèle ne concernait que les bonnes pratiques

[1] Actuellement ISO-15540.

du génie logiciel. Cependant, suite à un fort engouement, d'autres modèles ont vu le jour dans différents domaines tels que les ressources humaines avec le modèle People CMM ou l'ingénierie des systèmes avec SE-CMM. C'est pourquoi, CMM a été rebaptisé SW-CMM (CMM for Software).

CMM est un framework basé sur 5 niveaux de qualité successifs allant de 1 à 5 tels que le premier niveau qui contient les entreprises ayant peu ou aucune qualité dans leurs processus et pratiques logicielles et inversement pour le dernier niveau. Ces 5 niveaux sont : Initial - Répétable - Défini - Optimisé - Managé. Chaque niveau, excepté le premier, contient un ensemble de processus[2] appelé KPA (Key Process Area). Pour passer à un niveau supérieur, l'entreprise doit satisfaire à tous les objectifs de tous les KPA du niveau auquel elle se situe.

Chaque KPA est organisé en un ensemble de dispositifs communs qui décrivent l'infrastructure et les activités qui contribuent le plus à une implémentation efficace et à une institutionnalisation des KPA. Chaque pratique clé consiste en une simple phrase, souvent suivie par une description plus détaillée, pouvant inclure des exemples. Les dispositifs communs contiennent des pratiques clés qui sont des attributs indiquant si l'implémentation et l'institutionnalisation des KPA est efficace, répétable et durable (Descriptif adapté de [9]). Cette structure est décrite par la figure suivante [26] :

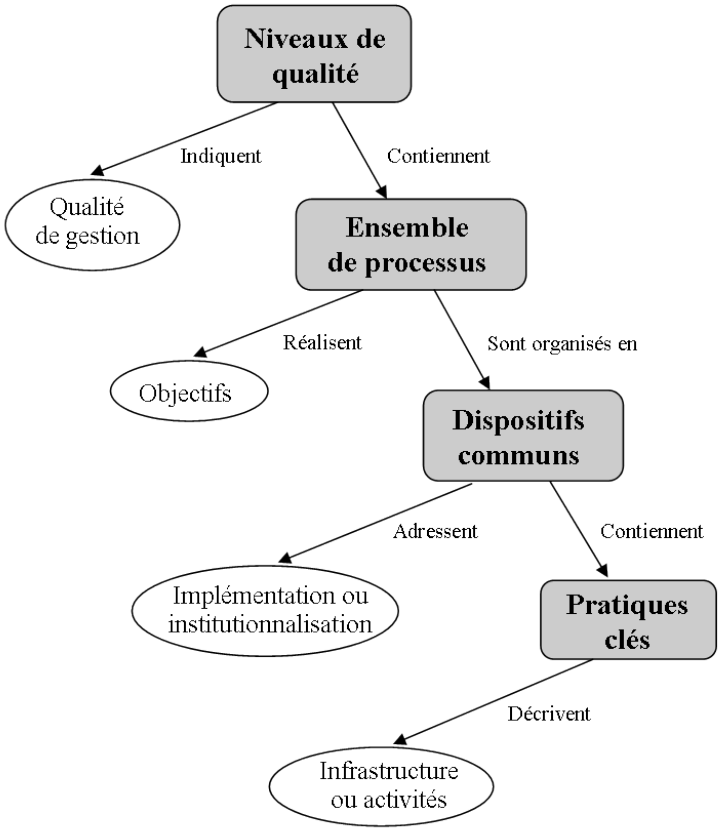



FIG. 1.1 – Architecture du modèle CMM

[2] *"Un processus est un ensemble structuré de pratiques nécessaires à la réalisation d'un objectif commun clairement défini"* [17]

*SPICE*

Le second modèle est SPICE (ISO 15504). Selon [22], SPICE est une norme créée par l'ISO pour standardiser l'évaluation des processus logiciels. C'est d'avantage un outil d'évaluation de la maîtrise de conduite du projet qu'une méthodologie de travail. SPICE est un référentiel des pratiques clés destiné à tout projet de développement ou de maintenance du logiciel. La première version de ce modèle est sortie en 1995.

SPICE fournit un framework pour l'évaluation des processus logiciels. Il peut être utilisé par les organisations travaillant dans les domaines du management, du monitoring, du développement,... de logiciels. L'architecture du modèle est constituée de 9 parties allant de 1 à 9 tel que le montre la figure 1.2 (Image tirée de [23] et adaptée via [24]). La partie 2 contient un modèle assez proche de celui de SW-CMM qui est composé de deux dimensions (Descriptif adapté de [9]) :

1. **Une dimension processus :** Cette dimension est composée du contenu des processus, de leurs objectifs mesurables et leurs résultats attendus, attestant de leur exécution. Elle est organisée en 5 catégories. Une catégorie est une ensemble de processus adressant le même espace d'activité. Les 5 catégories sont : client-fournisseur, ingénierie, support, management, organisation.

2. **Une dimension qualité des processus :** C'est une série d'attributs des processus représentant les caractéristiques mesurables nécessaires à leur gestion et leur amélioration. On peut alors classer les processus sur 6 niveaux de maturité : Incomplet - Exécuté - Géré - Établi - Prévisible - Optimisé.

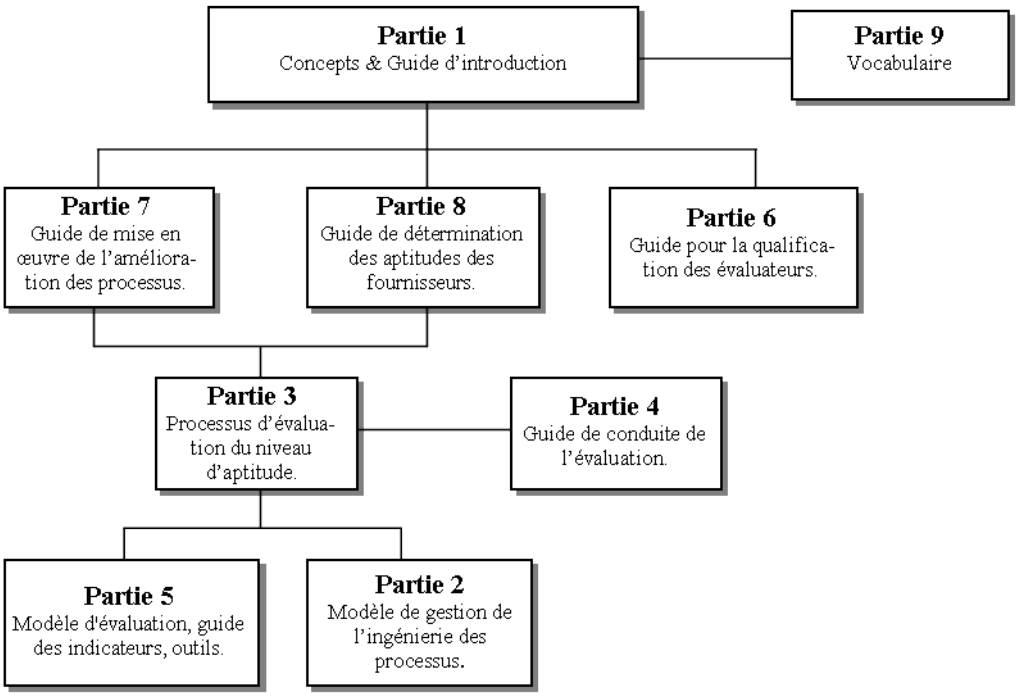



FIG. 1.2 – Architecture du modèle SPICE

Bien d'autres modèles existent à côté de CMM et SPICE. Par exemple, le projet CMM Integration qui a été élaboré pour mettre au point un premier ensemble de modèles intégrés. Il est différent de CMM au niveau de sa structure et de sa terminologie. Un autre exemple est Trillium qui est adapté au domaine des télécommunications. C'est une compilation de pratiques recueillies dans divers standards et modèles. Sa structure est très proche de SW-CMM 1.1. Un dernier exemple est Bootstrap qui est une initiative européenne, proche de SPICE au point de vue structure. Ce projet a été abandonné à l'heure actuelle.

L'objectif de ce mémoire étant basé sur les petites et très petites entreprises, le modèle développé ci-dessous est celui le mieux adapté à leur contexte.

## 1.3 Modèle et méthode OWPL

Comme le dit Anabel Stambollian [7], *"l'intérêt pour une entreprise ou organisation du monde du logiciel d'entreprendre une démarche de « qualité » n'est plus à démontrer. En effet, ce n'est que via cette démarche qu'elle pourra continuer à survivre et à progresser sur ce type de marché."* C'est pourquoi, des modèles tels que CMM ou SPICE se sont fortement propagés et sont en continuelles progression et amélioration. Cependant, ceux-ci ne ne visent que les grandes entreprises suite à la lourdeur de leur application, aux coûts qu'ils engendrent, aux ressources qu'ils consomment,... Adapter de telles approches s'avère être une tâche complexe et demande beaucoup d'expertise et de ressources financières : il faut d'abord comprendre le modèle que l'on désire adapter, puis l'adapter à l'entreprise souhaitée et ensuite l'implanter au sein de celle-ci et en mesurer les effets. Qu'en est-il alors des petites entreprises ne pouvant s'offrir de tels moyens ?

### 1.3.1 Présentation générale

Depuis plusieurs années maintenant, le Laboratoire de Qualité Logiciel (LQL) de L'Université de Namur (Belgique) a développé une pratique légère d'amélioration des processus et pratiques logiciels pour les petites entreprises : le modèle OWPL ([5], [6] ). Ce modèle a été développé entre les années 1998 et 2000 sur base de plusieurs approches mondialement connues tels SPICE ou CMM.

L'objectif du modèle OWPL est d'aider les entreprises à améliorer leurs pratiques logicielles en tenant compte d'une part des caractéristiques propres à leur structure et d'autre part des objectifs opérationnels et stratégiques de l'entreprise. Dans cette perspective, le modèle OWPL peut être considéré comme un référentiel proposant une liste de bonnes pratiques qu'il est bon de respecter si l'on veut limiter les risques associés à la mise en oeuvre des projets informatiques [17].

Le modèle OWPL n'impose pas la rédaction de procédures pour toutes les activités ni l'enregistrement systématique des résultats. Il recommande un niveau de formalisme approprié qui devra être défini en fonction de la structure de l'unité concernée et du type d'activités [17]. Ainsi, les entreprises seront en mesure d'améliorer la qualité de leurs processus et pratiques logiciels tout en n'étant pas submergées par des procédures superflues ou des tâches administratives non nécessaires et en utilisant le moins possible de ressources de l'entreprise.

Le contexte d'utilisation du modèle OWPL (Description adaptée de [18]) ainsi que sa structure (Description adaptée de [5], [6], [18]) vont maintenant être brièvement présentés.

### 1.3.2 Contexte d'utilisation

Le modèle OWPL repose sur l'hypothèse que toute activité d'une entreprise doit être réalisée afin d'atteindre l'objectif global de l'entreprise. Cette hypothèse est-elle fondée ? Une entreprise contient un ensemble de départements permettant ainsi de diviser son objectif général en sous-objectifs attribués aux différents départements. Un département regroupant différents processus, on peut donc diviser l'objectif du département en sous-objectifs attribués aux différents processus œuvrant dans le département. Vu qu'un certain nombre d'activités sont requises pour réaliser correctement un processus, on peut encore diviser l'objectif du processus en sous-objectifs attribués à ces différentes activités nécessaires (voir figure 1.3.2). L'hypothèse du modèle OWPL est donc bien fondée.

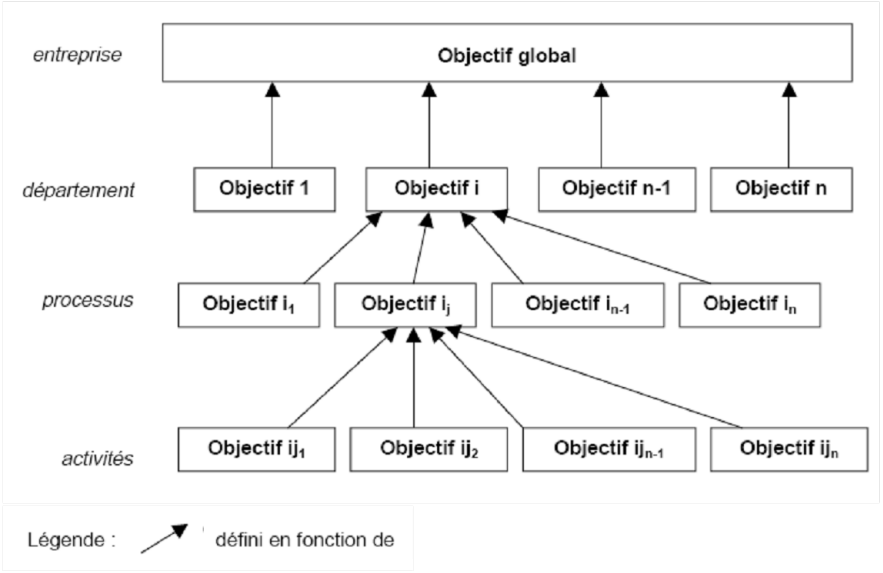



FIG. 1.3 – Tableau des objectifs

Cette description des objectifs est essentielle à toute démarche d'amélioration. En effet, elle permet de cibler les efforts, de motiver les acteurs et de contrôler l'efficacité de la démarche.

Pour garantir le succès de cette démarche et afin de stabiliser l'environnement d'exécution, certains facteurs, appelés "facteurs de succès" sont à prendre en compte. Ces facteurs sont, par exemple, la mise à disposition de ressources ou encore l'engagement de la direction.

### 1.3.3 Structure

Comme cité précédemment, la majorité des entreprises criblées par OWPL sont des petites entreprises ayant peu ou aucune expérience dans l'amélioration des processus et pratiques logiciels et se situant à un niveau de maturité faible. Les faiblesses présentes dans ces entreprises ne concernent généralement qu'un nombre restreint de processus de telle façon qu'il est possible de généraliser des bonnes pratiques logicielles qui pourront s'appliquer à d'autres entreprises.

Le premier objectif de la méthodologie est de faire prendre conscience aux petites entreprises de l'importance de l'amélioration des pratiques logicielles. Une fois cet objectif atteint, on passe alors à un deuxième objectif : celui d'initier un mécanisme continu d'amélioration des pratiques logicielles au sein de l'entreprise. Ce mécanisme d'amélioration doit apporter des résultats rapides et amenant une grande valeur ajoutée à l'entreprise tout en lui demandant l'effort minimal.

La démarche est graduelle (voir figure 1.3.3) et se base sur un cadre "framework" d'amélioration des pratiques logicielles, composé de trois étapes (figure adaptée de [6]). Notons que cette démarche n'est pas linéaire : les entreprises s'attardent au niveau le plus approprié selon leur taille et leur niveau de maturité actuel.

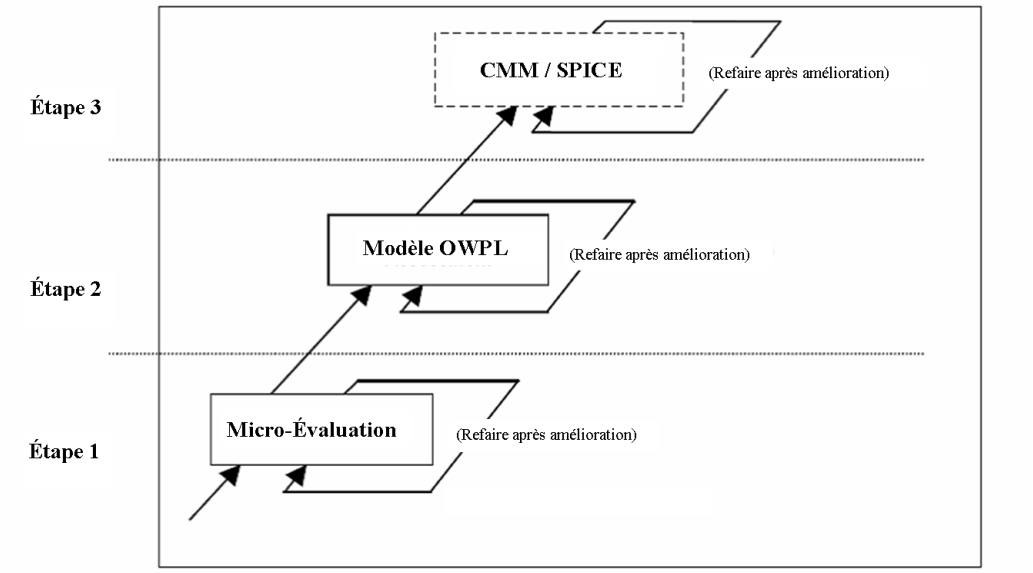



FIG. 1.4 – Démarche graduelle d'OWPL

Les différents niveaux représentés par la figure 1.3.3 sont, ci-dessous, brièvement développés.

### Micro-Évaluation :

Ce premier niveau[3] a pour objectif de dresser un bilan très général des pratiques logicielles d'une entreprise pour mettre en évidence les bonnes pratiques déjà présentes au sein de celle-ci ainsi que celles offrant des opportunités d'amélioration. Le but n'est donc pas de décrire dans le détail un secteur particulier mais de dresser une évaluation selon six axes fondamentaux, à savoir :

1. Gestion de la qualité,
2. Relations avec le client,
3. Relations avec les sous-traitants,
4. Développement et gestion de projet,
5. Gestion de produit,
6. Formation et gestion des ressources humaines.

La micro-évaluation se base sur un large questionnaire permettant de cibler les forces et faiblesses, les opportunités et risques ainsi que les recommandations à court et à long terme de l'entreprise évaluée, sur base d'un score attribué à chaque question. La réponse à ce questionnaire se réalise par le biais d'une interview d'environ une heure entre un représentant de l'entreprise et la personne en charge de l'évaluation.

### Évaluation OWPL :

Le modèle OWPL est le composant central de cette méthodologie. Comme dit précédemment, ce modèle est basé sur le contexte particulier des petites entreprises et sur certains modèles de processus existants tels que CMM ou SPICE. L'évaluation du processus est basée sur le rapport de la micro-évaluation ainsi que les buts de l'entreprise. Le modèle utilise un vocabulaire simple et propose l'amélioration d'un nombre réduit de processus et pratiques.

OWPL est composé d'un ensemble de pratiques et facteurs de succès (voir figure 1.3.3). Il définit 10 processus donc chacun est composé d'un ensemble de pratiques logicielles devant être suivies afin de remplir au mieux le processus considéré. Ces processus sont les suivants :

1. Gestion des exigences,
2. Planification,
3. Suivi et supervision de projet,
4. Développement,
5. Documentation,
6. Tests,
7. Gestion de configuration,
8. Gestion des sous-traitants,
9. Gestion de la qualité,
10. Capitalisation des acquis.

[3] décrit plus précisément dans la section 2.4.

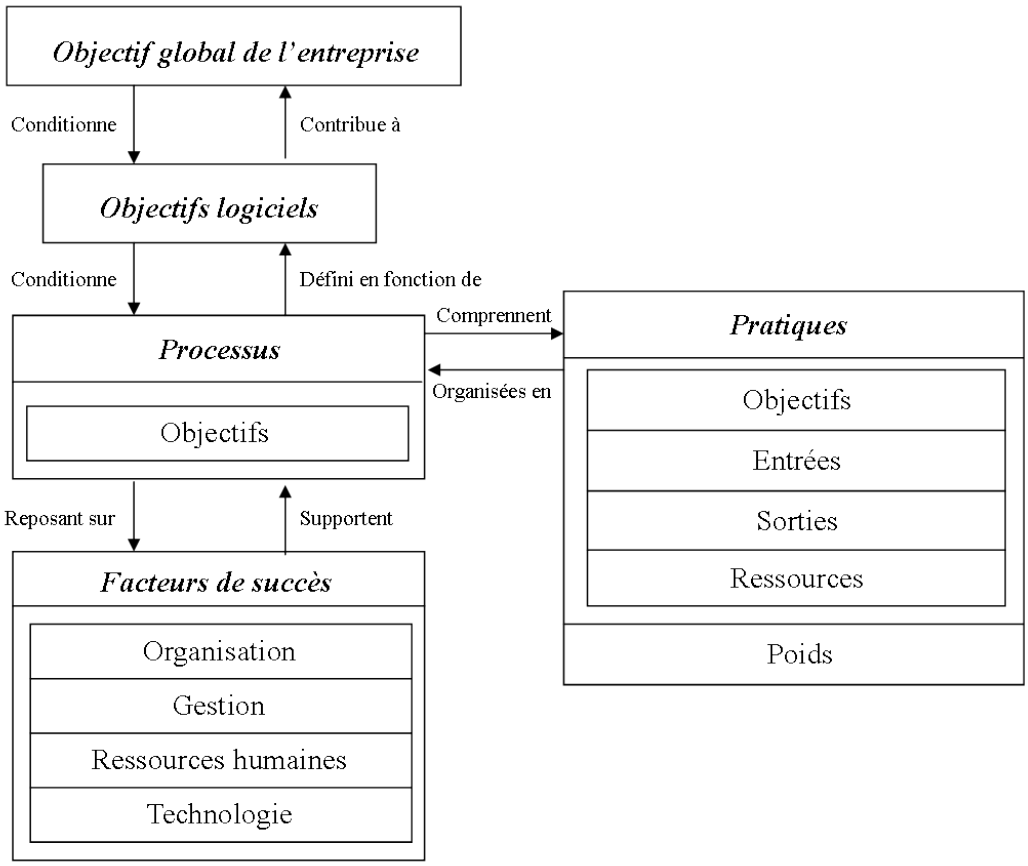



FIG. 1.5 – Structure du modèle OWPL

Un processus est *"un ensemble structuré de pratiques nécessaires à la réalisation d'un objectif commun clairement défini"* [18]. Dans le cadre d'OWPL, un processus est défini en fonction de l'objectif de l'entreprise, duquel découle son objectif propre. Afin d'atteindre cet objectif, le processus est défini par un ensemble de pratiques et s'appuie sur un certain nombre de facteurs de succès.

*Les pratiques*

Une pratique est *"une activité d'ingénierie qui contribue à la réalisation de l'objectif d'un processus par la création d'un délivrable*[4] *ou l'amélioration de la capacité du processus"* [18]. Chaque pratique possède un objectif, des entrées, des sorties et des ressources allouées. Afin de déterminer quelles pratiques doivent être réalisées en premièr, celles-ci comportent chacune un poids indiquant leur importance dans la réalisation des objectifs du processus.

[4]Un délivrable est un élément concret produit durant le déroulement d'un projet.

*Les facteurs de succès*

Les facteurs de succès sont *"des éléments d'environnement qui favorisent la mise en place d'un support permettant une exécution optimale des processus"* [18]. Comme le montre la figure 1.3.3, ils sont regroupés en 4 catégories

1. l'organisation dans laquelle les processus sont en place,
2. la politique de gestion mise en place,
3. les ressources humaines mobilisées et
4. les moyens techniques utilisés.

De même que pour la micro-évaluation, l'évaluation OWPL travaille avec un questionnaire permettant d'établir un rapport d'évaluation. Pour évaluer la qualité des processus et pratiques logicielles dans l'entreprise, le questionnaire travaille aussi avec un système de cotation.

### Évaluation CMM ou SPICE :

Si la taille de l'entreprise ou le contexte dans lequel elle évolue justifie la nécessité d'une certification et si elle a atteint un niveau de maturité déterminé, une évaluation basée sur le modèle SPICE ou CMM peut être réalisée. La démarche utilisée pour cette évaluation dépend du modèle de référence.

## 1.4 Conclusion

La méthodologie associée au modèle OWPL semble être la meilleure éventualité au problème d'amélioration de la qualité des processus et pratiques logiciels dans les petites et très petites entreprises puisque CMM ou SPICE n'y sont pas adaptés. Cependant, ce travail s'arrêtera au premier niveau de la figure 1.3.3, à savoir : la micro-évaluation. Cette méthode est décrite dans le chapitre suivant.

# Section 2 :

## *La micro-évaluation*

**Contenu :** *Description détaillée de la méthode de micro-évaluation d'OWPL.*

### 2.1 Introduction

L'industrie du logiciel devenant de plus en plus compétitive, il semble logique que les entreprises la composant implantent de plus en plus des modèles, des normes et des référentiels. Cependant, il est nécessaire d'évaluer les résultats de ces implantations. Dans ce but, certaines méthodologies ont été mises en place. Parmi les plus connues :

- **SCAMPI** [13] (Standard CMMI Assessment Method for Process Improvement) qui permet de déterminer le niveau de maturité CMMI d'une entreprise
- **RAPID** [14] (Rapid Assessment for Process Improvement for software Development), basé sur SPICE, qui permet d'évaluer les processus et pratiques logiciels d'une entreprise en une journée.

La méthode présentée ici est celle de la micro-évaluation. Cela se fera en trois étapes : la caractérisation des entreprises ciblées par une telle méthode,l'énonciation des concepts et hypothèses cachés derrière celle-ci et la présentation de la méthodologie.

### 2.2 Entreprises ciblées par la micro-évaluation

La micro-évaluation peut être utilisée par les petites et moyennes entreprises ou les très petites entreprises. Ces types d'organisation désignent les entreprises de taille modeste, par rapport, notamment, à leur nombre de salariés ou leur chiffre d'affaires. Pierre-André Julien et Michel Marchesnay en ont identifié un ensemble de caractéristiques communes [21] :

1. Petite taille ;
2. Centralisation et personnalisation de la gestion autour du propriétaire-dirigeant ;
3. Faible spécialisation du travail ;
4. Stratégie intuitive ou peu formalisée, forte proximité des acteurs ;
5. Système d'information interne simple et peu formalisé ;
6. Système d'information externe simple basé sur les contacts directs.

En plus des différentes caractéristiques énoncées ci-dessus, et sur base de celles-ci, certaines caractéristiques implicites peuvent encore être rajoutées, à savoir :

1. Proximité entre patron et employés ;
2. Faible formalisation ;
3. Le recours à l'écrit n'est pas primordial, du fait de l'importance de l'ajustement mutuel ;
4. Structure plate ;
5. Pas de niveaux hiérarchiques, ou très peu.

Comme le souligne [21], elles jouent un rôle primordial dans la création d'emploi dans de nombreux pays. En France par exemple, 99% des entreprises françaises sont des PME : 92% des très petites entreprises (TPE) et 7% de petites entreprises (PE). De plus, les innovations sont souvent le fait de PME.

## 2.3 Concepts derrière la micro-évaluation

L'objectif de la micro-évaluation est l'amélioration de processus et pratiques logicielles d'une entreprise. Afin d'atteindre au mieux cet objectif, certaines hypothèses sont à prendre en compte lors de l'utilisation de cette méthodologie simple et légère. Ces hypothèses sont les suivantes ([9] et [18]) :

1. Les entreprises pour lesquelles cette méthode peut être appliquée doivent être des petites ou moyennes entreprises ne comportant au maximum que 120 employés ou de très petites entreprises de maximum 10 employés [5]. L'avantage de ne travailler qu'avec de telles entreprises permet de prendre en considération leur contexte. Ainsi, l'objectif n'est pas d'appliquer une méthodologie type mais d'adapter une méthodologie à l'entreprise en question afin d'apporter les résultats réalisant le plus de valeur ajoutée pour celle-ci. Une dernière remarque est que les entreprises évaluées ont peu, voire aucune expérience dans l'amélioration des processus et pratiques logicielles. De plus, elles se situent à un niveau de maturité faible.

2. Étant confronté à des petites entreprises, le vocabulaire à utiliser doit être le plus simple possible. De plus, celui-ci doit se composer d'un nombre infime de termes techniques . Il doit rester le plus accessible possible puisque les personnes interviewées ne possèdent pas obligatoirement des connaissances poussées en informatique.

3. L'approche utilisée par la micro-évaluation doit être progressive. En effet, lors de chaque déclenchement d'un processus d'amélioration, l'objectif est d'améliorer un point de l'entreprise et non tout le processus en une fois. Il est donc nécessaire, avant tout déclenchement, de valider les problèmes à résoudre avec les dirigeants de l'entreprise c'est-à-dire donner une priorité aux problèmes afin de déterminer ceux qui doivent être résolus dans les premiers cycles d'amélioration.

4. Les labels et niveaux de maturité ne sont pas important. Tout comme la documentation générée, celle-ci doit rester la plus légère et la plus simple possible.

Avant de passer à la micro-évaluation proprement dite, il est nécessaire de s'interroger sur les réelles motivations des petites et très petites entreprises à entreprendre une démarche d'amélioration de leurs processus et pratiques logiciels. La motivation principale réside dans la recherche d'une meilleure efficacité de production et d'une meilleure qualité de produit. En effet, les petites et très petites entreprises ne possèdent pas les ressources nécessaires pour décrire complètement et formellement les exigences du client, pour entreprendre des phases de revue de code bien planifiées ou pour planifier correctement leurs projets. Par une démarche d'amélioration, ces entreprises travailleront avec des ressources externes qui vont développer une application qui aidera ces entreprises à améliorer leurs pratiques et processus déjà en place.

D'autres motivations viennent s'ajouter à celle décrite ci-dessus : les faibles coûts d'une telle démarche, la rapidité du processus d'amélioration se déroulant entre 2 et 3 mois, la simplicité de la méthode puisqu'elle ne travaille qu'avec un questionnaire ( afin d'établir la situation actuelle de l'entreprise) ou encore le peu de temps que les dirigeants de l'entreprise ont à consacrer à cette démarche d'amélioration.

La description de la méthode pouvant être utilisée dans le but d'améliorer les processus et pratiques logiciels d'une entreprise va maintenant être passée en revue.

## 2.4 La micro-évaluation

La micro-évaluation se base sur un questionnaire simple permettant de collecter un ensemble d'informations sur les pratiques logicielles en cours dans l'entreprise évaluée ainsi que celles qui offrent des opportunités intéressantes. Elle se réalise en une durée d'une heure environ via une entrevue entre une personne de l'entreprise évaluée et la personne en charge de l'évaluation. Le questionnaire utilisé couvre 6 axes clés. Le choix de ces axes est réalisé sur base de l'expérience acquise lors d'évaluations précédentes dans différentes PME et correspondent à des secteurs-clés des niveaux 2 et 3 de CMM ([5]). Les 6 axes sont :

1. Gestion de la qualité,
2. Relation avec le client,
3. Relation avec le fournisseur,
4. Management du développement et du projet,
5. Management du produit,
6. Formation - Gestion des ressources humaines.

### 2.4.1 Support de la micro-évaluation

Le questionnaire contient 16 questions couvrant les 6 axes préalablement cités. Chaque question possède des sous-questions afin d'affiner les réponses données par l'entreprise. Jusqu'à présent, différentes versions du questionnaires ont été établies, chacune étant une amélioration de la précédente. La version reprise ci-dessous correspond à celle utilisée au Québec (Canada) et est celle utilisée durant le stage (repris de [10])[1] :

[1]Les sous-questions n'ont pas été reprises pour une meilleure lisibilité

1. **Introduction**
   (a) Pouvez-vous me présenter brièvement votre société, ses activités, ses principaux acteurs (employés, clients, fournisseurs,...) ?
   (b) Pouvez-vous me décrire le contexte dans lequel évolue votre service à l'heure actuelle ? Quels sont les principaux projets en cours, quelles sont les priorités, etc... ?
2. **Gestion de la qualité**
   (a) Est-ce que votre entreprise a déjà entamé des activités qualités ?
   (b) Quels moyens mettez-vous en oeuvre pour garantir la qualité des produits logiciels que vous développez ?
3. **Relation Clients**
   (a) Formalisez-vous les exigences de vos clients ?
   (b) Comment les clients vous font-ils part des modifications à apporter aux fonctionnalités attendues ?
   (c) Organisez-vous des réunions de coordination avec vos clients en cours de projet ?
4. **Gestion des Sous-traitants**
   (a) Comment assurez-vous la sélection de vos sous-traitants ?
   (b) Comment assurez-vous le suivi du travail de vos sous-traitants ?
5. **Développement et Gestion de projet**
   (a) Découpez-vous vos projets en phases/étapes ?
   (b) Suivez-vous une méthodologie de développement pour vos projets ?
   (c) Vos développements font-ils l'objet de plannings ?
   (d) Pouvez-vous connaître à tout moment l'état d'avancement et les ressources consommées par chaque projet en cours ?
   (e) Les membres de l'équipe se réunissent-ils régulièrement pour analyser les problèmes survenant en cours de projet ?
   (f) Comment faites-vous pour détecter et corriger les erreurs en cours de projet ?
6. **Gestion des produits**
   (a) Les produits de votre travail sont-ils identifiés de manière systématique ? Les différentes versions sont-elles gérées et archivées ?
   (b) Utilisez-vous des documents types, des structures prédéfinies pour les produits de votre travail ?
7. **Formation et Gestion des ressources humaines**
   (a) Existe-t-il une politique de formation ?

Comme l'expliquent [8] et [11], il existe deux types de questions : Premièrement, celles concernant les pratiques générales de l'entreprise et deuxièmement, celles concernant les pratiques logicielles. Les premières sont cotées sur une échelle linéaire en fonction du niveau de qualité de la pratique tandis que les deuxièmes sont cotées en référence à une grille à deux entrées en fonction du niveau de qualité de la pratique ainsi que de son degré d'institutionnalisation effectif au sein de l'organisation. Les deux types de question utilisent des échelles de valeur allant de 0 (pratique absente) à 4 (pratique excellente). Les deux figures suivantes représentent ces deux types de questions :

**Est-ce que votre entreprise a déjà entamé des activités qualités ?**

| Proposition | Rép. | Points |
|---|---|---|
| Non | | 0 |
| Actions qualité ponctuelles | x | 1 |
| Procédures suivies par toutes les équipes | | 2 |
| Procédures utilisées et adaptées en permanence | | 4 |

FIG. 2.1 – Question basée sur les pratiques générales de l'entreprise

**Formalisez-vous les exigences de vos clients ?**

| | Projets | |
|---|---|---|
| Proposition | Certains | Tous |
| Oui, en interne (unilatéral) | Score = 1 | Score = 2 |
| Oui, avec approbation du client | Score = 2 | Score = 4 |

FIG. 2.2 – Question basée sur les pratiques logicielles de l'entreprise

### 2.4.2 Résultat de la micro-évaluation

Avec les résultats de l'interview, un rapport de micro-évaluation est généré. Celui-ci contient environ une douzaine de pages. Il commence par une introduction contenant une description de l'environnement de l'entreprise ainsi que les lignes directrices de la micro-évaluation. Ensuite, on retrouvera les résultats du questionnaire selon les 6 axes. A la suite, on retrouvera l'ensemble des forces et faiblesses de l'entreprise, les opportunités et risques qu'elle peut rencontrer ainsi que les recommandations à court et long terme établies par l'équipe d'évaluation (Éléments retirés et adaptés de [11] et [6]). Le plan type d'un rapport de micro évaluation est donc le suivant :

1. Introduction
2. Résultats de l'interview selon chaque axe
3. Forces et faiblesses
4. Risques et opportunités
5. Recommandations à court et long terme

A la fin du rapport, se placent deux tableaux synoptiques représentant graphiquement les résultats du questionnaire préalablement écrit dans le rapport. Leur unique interprétation ne saurait suffire à avoir une vue objective et correcte sur les pratiques logicielles en place dans l'unité évaluée. Les deux schémas ci-dessous représentent ces tableaux synoptiques. Le premier

représente les résultats du questionnaire de façon résumée tandis que le deuxième les représente de façon détaillée. Voici deux exemples pour illustrer ces deux tableaux [2].) :

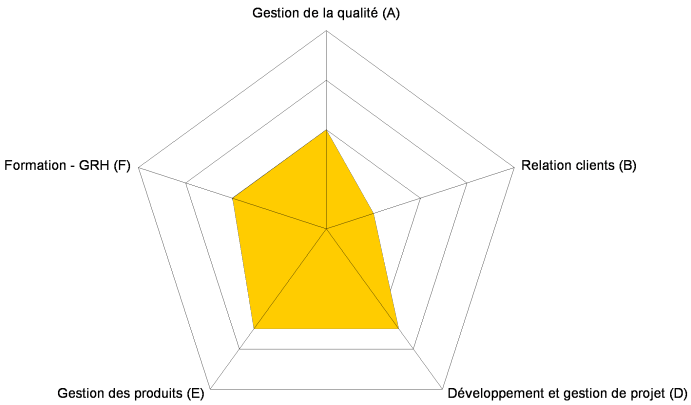



FIG. 2.3 – Exemple de tableau synoptique résumé

Dans le cas du tableau synoptique résumé, la surface colorée est le résultat de la projection de la moyenne des scores obtenus pour chaque question située sous la même rubrique, "Relation Clients" et "Gestion des sous-traitants" ne formant qu'une seule rubrique[3].

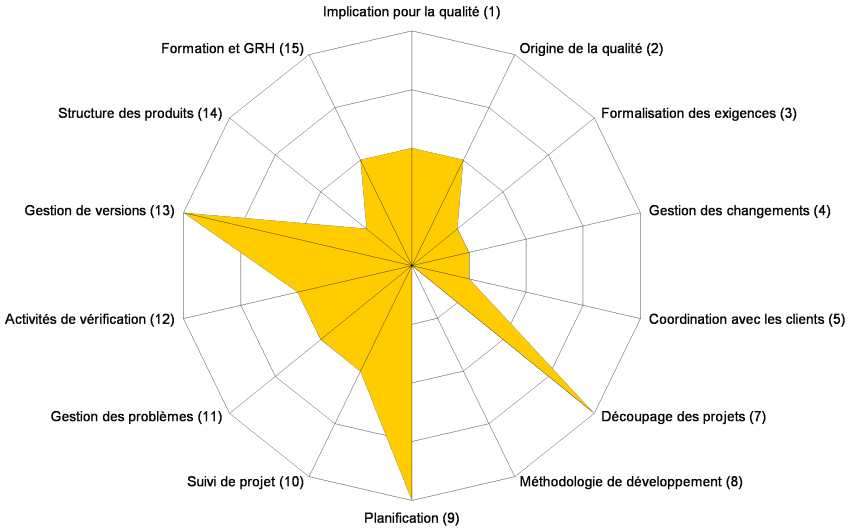



FIG. 2.4 – Exemple de tableau synoptique détaillé

[2]Les tableaux synoptiques sont repris du rapport de micro-évaluation réalisé le 20 mai 2006 à l'ETS sur Club Capra

[3]La section "Introduction" du questionnaire n'y étant pas reportée.

Pour le tableau synoptique détaillé, la surface colorée est le résultat de la projection des scores obtenus pour chaque question composant le questionnaire de la micro-évaluation[4].

### 2.4.3 Confidentialité du rapport

Comme il est précisé dans [11], le rapport est confidentiel et envoyé individuellement aux personnes ayant pris part à l'interview. C'est pourquoi, celui-ci contient l'accord de confidentialité suivant : *"Les informations sont traitées de manières confidentielles. Les résultats de la micro-évaluation seront uniquement communiqués à la personne contactée dans le cadre de cette évaluation. Les statistiques consolidées de ces informations avec celles d'autres sociétés seront systématiquement rendues anonymes"*[1].

Le dernier point qui sera abordé concerne la relation entre les différents axes de la micro-évaluation et quelques secteurs de processus de CMMI.

## 2.5 Lien entre la micro-évaluation et CMMI

Comme spécifié précédemment, la micro-évaluation et son questionnaire sont fortement inspirés des modèles tels que SW-CMM ou SPICE. Ainsi, il est possible d'établir une forte relation entre les 6 axes clés de la micro-évaluation et des secteurs de processus de CMMI [12]. Cependant, le degré de détails est différent de CMMI en termes d'objectifs de la micro-évaluation [11]. Cette liaison est illustrée par le tableau suivant [11] :

| **Axes de la micro-évaluation** | **Secteurs de processus de CMMI** |
|---|---|
| Gestion de la qualité | Assurance qualité produit et processus |
| Relation avec le client | Gestion et développement des exigences |
| Relation avec le fournisseur | Gestion des accords avec les fournisseurs |
| Management du développement et du projet | Solution technique & Planning de projet, Contrôle et monitoring de projet |
| Management du produit | Gestion de configuration |
| Formation - GRH | Formation organisationnelle |

FIG. 2.5 – Liens entre la micro-évaluation et CMMI

## 2.6 Conclusion

Après cette description détaillée de la micro-évaluation et de la méthodologie qui lui est associée, il semble clair que celles-ci sont les plus adaptées aux petites et très petites entreprises. Ce succès est dû à plusieurs facteurs : Premièrement, l'approche utilisée est progressive permettant ainsi à l'entreprise de ne pas devoir allouer de nombreuses ressources à la démarche

[4]La section "Introduction" du questionnaire n'y étant pas reportée.

d'amélioration. Deuxièmement, le processus d'amélioration est basé sur les objectifs de l'entreprise et sur les résultats obtenus par le questionnaire de la micro-évaluation, ce qui permet d'améliorer une faiblesse importante de l'entreprise. Enfin, il peut aussi être cité la compatibilité avec des modèles tels que CMM ou SPICE.

# Section 3 :

## *Démarche d'amélioration*

> **Contenu :** *Présentation de la méthode utilisée afin d'entreprendre au mieux les différentes démarches d'amélioration.*

### 3.1 Introduction

Bien que la micro-évaluation soit une méthode performante et rapide pour évaluer la qualité des processus et pratiques logicielles dans les petites et très petites entreprises, il est nécessaire d'opter pour une démarche d'amélioration permettant d'utiliser au mieux la micro-évaluation dans les différentes interventions faites en entreprises. Afin de ne pas utiliser la première démarche venue, différents aspects sont à prendre en compte. La démarche doit être facile à mettre en oeuvre, facilement adaptable et d'une simplicité équivalente à celle de la micro-évaluation. Un dernier aspect à ne pas sous-estimer est que la micro-évaluation permet de faire ressortir différents problèmes ayant traits à différentes phases d'un développement logiciel. C'est pourquoi, la démarche doit être cyclique, impliquant ainsi que chaque cycle se focalise sur une démarche d'amélioration possible dans l'entreprise ciblée.

Sur base de ses différentes recommandations, différentes démarches ont été entreprises sur le principe de base de la Roue de Deming.

### 3.2 Roue de Deming

La roue de Deming représente un cycle d'amélioration de la qualité. Cette méthode est également connue sous l'acronyme PDCA (**P**lan **D**o **C**heck **A**ct), des quatre lettres qui en définissent les étapes. Son nom vient du statisticien William Edwards Deming. Cette méthode permet de maîtriser et d'améliorer un processus par l'emploi d'un cycle en quatre étapes visant à améliorer la qualité d'un produit ou d'un service. Comme le montre la figure ci-dessous (tirée de [29]), chaque étape sert de base à la suivante, la fin d'une étape entraînant automatiquement le démarrage de sa suivante [28, 29]).

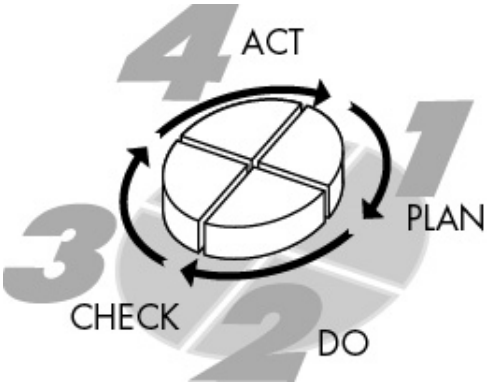



Fig. 3.1 – Roue de Deming

La roue de Deming est composée de 4 étapes, à savoir (tiré et adapté de [28, 29]) :

1. **Plan :** Cette étape définit l'objectif. Durant le "plan", on identifie et on précise les besoins du bénéficiaire du produit. On inventorie les moyens nécessaires à sa réalisation, son coût et son planning. Cette première étape est donc une étape de planification et d'écriture du cahier des charges relatif à l'amélioration future qui sera apportée dans l'entreprise (produit, méthodologie,...).

2. **Do :** C'est l'étape chantier c'est-à-dire l'étape durant laquelle, sur base du cahier des charges et de la planification, on va entreprendre la réalisation du produit souhaité.

3. **Check :** Durant de cette troisième étape, on vérifie si le travail réalisé pendant la phase "do" correspond aux besoins exprimés durant la phase "Plan", et ce dans les délais et coûts précisés durant cette même première phase. Cette étape utilise des moyens de contrôle divers comme par exemple des indicateurs de performance.

4. **Act :** La dernière étape consiste à chercher les améliorations à apporter au projet. L'étape Act amènera un nouveau projet à réaliser, donc une nouvelle planification à établir.

N'oublions pas que la roue de Deming vise à améliorer un produit de façon continue en déclenchant divers cycles d'amélioration successifs. Ainsi, cette roue pourrait aussi être vue sous la forme d'une vis qui ne prendra fin qu'une fois la mort du produit arrivée. Le haut de la vis, le début du projet, est plus important par sa préparation, sauf si un besoin d'adaptation aux événements est nécessaire (tiré et adapté de [29]). Cette vision est schématisée par la figure suivante (tirée de [29]) :

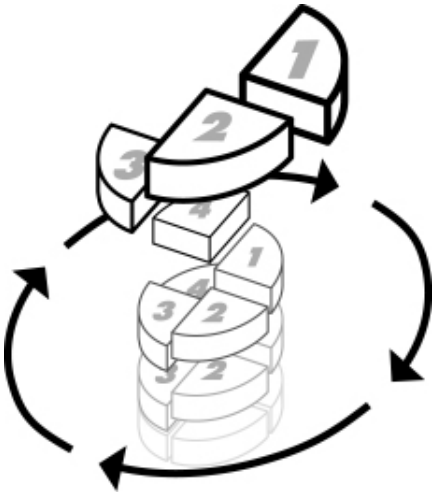



FIG. 3.2 – Roue de Deming

Afin d'établir une démarche d'amélioration utilisant la micro-évaluation, il est nécessaire d'appliquer les concepts-clés de la roue de Deming.

## 3.3 Application à la micro-évaluation

Avant tout déclenchement d'un cycle de démarches d'améliorations successives, il est nécessaire d'appliquer la méthode de la micro-évaluation sur l'entreprise ciblée afin d'en faire ressortir les différentes forces et faiblesses, opportunités et risques, ainsi que les différentes recommandations à court et long terme relatives à l'entreprise ciblée. Une fois le rapport de la micro-évaluation établi, un cycle d'améliorations successives, proche de la roue de Deming, peut être déclenché. Chaque cycle contient alors 5 phases, à savoir :

- **Sélection :** Cette première étape consiste tout d'abord à évaluer l'entreprise au moyen de la micro-évaluation. Une fois cette évaluation réalisée et son rapport élaboré, il est alors possible d'établir un plan d'action d'amélioration sur base de ce rapport. Ce plan d'action devra contenir les différentes démarches d'amélioration devant être réalisées pour l'entreprise, une prioritisation de celles-ci et une grille de sélection permettant de cibler la démarche la plus importante. Ce plan d'action se termine par la démarche d'amélioration sélectionnée sur base de la grille préalablement établie.
- **Élaboration :** Cette deuxième étape consiste à établir le cahier des charges relatif à la solution devant être implémentée afin d'entreprendre au mieux la démarche d'amélioration, sélectionnée lors de la première étape. De plus, l'élaboration d'une méthodologie, d'un template ou d'un processus peut s'avérer nécessaire pour l'utilisation future du produit.
- **Implémentation :** Cette troisième étape consiste à implémenter la solution sur base des différents éléments établis lors de la phase précédente.

– **Vérification et validation :** Cette avant-dernière étape consiste, quant à elle, à vérifier que le solution fonctionne correctement dans les situations normales et qu'il est le plus robuste dans les autres situations. De plus, il faut aussi vérifier que la solution répond bien aux exigences spécifiées dans le cahier des charges établi lors de la phase de l'élaboration et qu'il remplit bien les conditions établies dans le plan d'action de sélection.

– **Installation et Formation :** Cette dernière phase consiste à installer la solution dans les locaux de l'entreprise ainsi qu'à former les futurs utilisateurs de celle-ci.

Une démarche d'amélioration cyclique est ainsi obtenue et dans laquelle chaque cycle représente l'amélioration d'un des points repris dans le rapport de la micro-évaluation préalablement établi. Une fois un cycle terminé, le suivant peut démarrer sur base du même rapport de micro-évaluation que celui utilisé lors de la phase précédente ou sur base d'un nouveau rapport si une nouvelle évaluation de l'entreprise a été réalisée.

## 3.4 Conclusion

Sur base de la démarche élaborée précédemment, deux démarches d'amélioration complète ont été entreprises pour deux entreprises québécoises différentes. Il est cependant à remarquer que les rapports de micro-évaluation relatifs à ces deux entreprises avaient déjà été générés. Ainsi, les parties consacrées aux études de cas ont directement pu être déclenchées. Ces deux démarches vont sont présentées dans les deux parties suivantes : la première concerne un club d'étudiant de l'École de Technologie Supérieure de Montréal, œuvrant dans le secteur de la robotique, tandis que la deuxième concerne une entreprise informatique québécoise travaillant dans le domaine agro-environnemental.

# Deuxième partie

# Première étude de cas

# Section 1 :

## *Sélection de l'amélioration*

> **Contenu :** *Identification et analyse des différentes pratiques à améliorer chez Club Capra ainsi que la sélection d'une de celles-ci. Elle commence par présenter l'entreprise.*

### 1.1 Présentation de l'entreprise

La première entreprise étudiée fut un club d'étudiants de l'École de Technologie Supérieure de Montréal (ETS), Québec, Canada. Ce club, connu sous le nom de Club Capra, exerce dans le domaine de la robotique dynamique et autonome. Il a été créé il y a quelques années par des étudiants de maîtrise sur base d'un projet de fin d'étude. L'objectif d'un club d'étudiant est de réunir les différentes branches du génie afin de réaliser un projet qui passionne ses membres. Le club se compose d'une quinzaine d'étudiants de l'ETS venant de différents domaines tels que le génie logiciel principalement, le génie mécanique ou électrique.

Afin d'être reconnu comme club étudiant et d'obtenir des subsides de l'ETS, Club Capra doit participer à différentes compétitions et expositions. Les compétitions sont au nombre de deux et se déroulent chaque année. Le but premier du club est de participer à celles-ci avec l'idée d'être le plus performant possible, d'où son intéressement à l'amélioration de la qualité de son processus logiciel.

### 1.2 Problèmes présents dans l'entreprise

La première étape consiste à élaborer les différents problèmes relevés par le rapport de la micro-évaluation [1] réalisé sur Capra afin d'en établir un plan d'action. Ce rapport est réalisé dans le cadre du projet d'APPEQ[1] et permet d'établir les forces, les faiblesses, les opportunités et les risques, ainsi que les recommandations à court et long terme relatives à l'entreprise évaluée.

Le tableau synoptique de la figure 1.2 représente la situation de Club Capra selon les différents axes de la micro-évaluation. On peut constater que sur certains axes tels que la

[1] **A**mélioration de la **P**erformance des **P**etites **E**ntrerprises **Q**uébecoises

planification ou la gestion des versions, Club Capra est dans une situation plus que parfaite alors que pour les axes tels que la méthodologie de développement ou la gestion de la qualité de ses produits, l'organisation est dans une situation inverse.

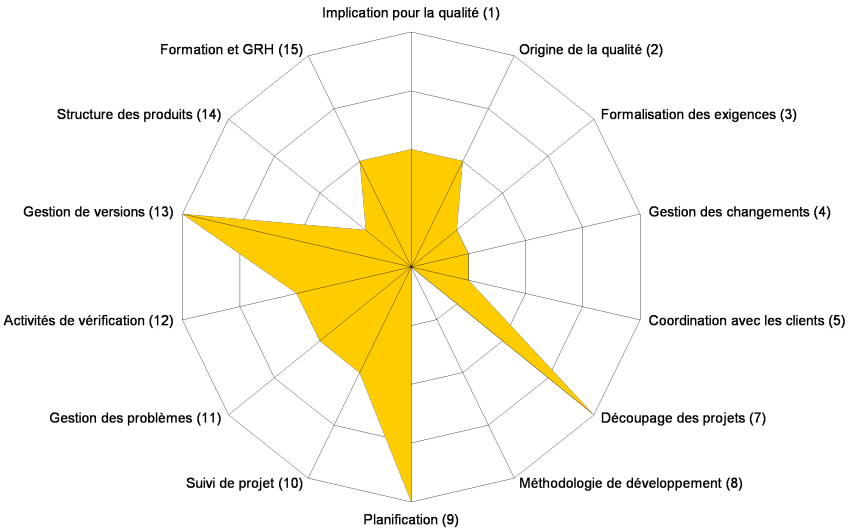



FIG. 1.1 – Synoptique détaillée de Capra

Les différentes pratiques à améliorer au sein de Club Capra sur base du rapport de la micro-évaluation sont [1] :

### Gestion de la qualité

Les problèmes relatifs à la gestion de la qualité chez Club Capra concernent la revue de code et les tests unitaires. En effet, la revue de code semble être le problème le plus important puisqu'il n'existe aucune méthodologie ni d'outil pour cette technique. C'est pourquoi, lors des différentes compétitions, il n'est pas possible pour ce club de mener à terme les épreuves obligatoires et donc, d'être le plus performant possible. Pour les tests unitaires, ceux-ci vont de pair avec le premier problème puisque le but de ces tests réside dans la détection d'erreurs d'implémentation.

### Gestion des changements

En seconde position, on retrouve des problèmes relatifs à la gestion des changements. Ce problème fait référence à du moyen terme et a trait à la formalisation des exigences. Participant à des compétitions, le club est sujet à des modifications d'exigences provenant du comité d'organisation. Ces changements peuvent arriver à tout moment avant la compétition. N'ayant pas de méthodologie pour les gérer, le club peut alors en omettre et ainsi, ne pas remplir les exigences demandées par la compétition.

### Processus de développement

De même que pour le problème précédent, le problème relatif au processus de développement se réfère à du moyen terme. En effet, les membres du club ne suivent pas un processus de développement bien formel pour chaque itération. Les étapes d'analyse des besoins et de validation et vérification du travail effectué sont minimes et n'apportent donc pas les résultats que le club pourrait espérer. Cette absence d'un processus formel de développement est sans doute dû au fait que Club Capra est un club et non à une entreprise et que celui-ci est composé d'étudiants surtout intéressés par les phases de codage.

### Gestion des risques

Le dernier problème décelé chez Club Capra via le rapport de la micro-évaluation concerne le gestion des risques. En effet, Club Capra n'en possède aucune suite au manque de temps pour réaliser des tests d'intégration. Il leur est donc impossible d'identifier les risques liés à leur projet, ce qui augmente donc la probabilité d'incompatibilité entre différents composants logiciel du produit final.

## 1.3 Critères de sélection de la pratique à améliorer

Une fois les différentes pratiques à améliorer répertoriées, il ne reste plus qu'à établir les critères de choix permettant d'éliminer celles qui ne peuvent être solutionnées. La partie du stage effectué au sein de Club Capra ne durant que 3 mois, le premier critère retenu fut celui du court terme. En effet, il n'était pas possible d'améliorer des pratiques relatives à du moyen terme (plus de 6 mois). Il semble donc évident que celles traitant de la méthodologie de développement et de la gestion des changements ne doivent pas être retenues.

Deux pratiques demeurent encore : celle relative à la gestion de la qualité, à savoir la revue de code et les tests unitaires et celle relative à la gestion des risques. L'objectif d'intervention faite lors du stage est d'améliorer une pratique logicielle de l'organisation via une solution peu coûteuse et qui amène une forte valeur ajoutée. Il est donc logique que celle liée à la gestion de risque ne soit pas retenue.

## 1.4 Choix de la pratique à améliorer

Grâce à l'énumération des différentes pratiques à améliorer sur base du rapport de la micro-évaluation ainsi que l'élaboration des critères de sélection, une proposition d'amélioration de leur gestion de qualité a été faite aux dirigeants de Club Capra. Principalement celles ayant trait à la revue de code et secondement, celles liés aux tests unitaires. Après une entrevue avec les dirigeants du club, la proposition a été acceptée et l'élaborer et l'implémenter d'une solution ont pu commencer. Cette amélioration sera donc le premier cycle d'amélioration des processus et pratiques logiciels chez Club Capra.

# Section 2 :

## *Premier cycle d'amélioration*

> **Contenu :** *Description de l'élaboration de la solution d'amélioration relative à la gestion de la qualité c'est-à-dire la phase d'analyse nécessaire à sa bonne implémentation.*

### 2.1 Recherche d'une solution d'amélioration

La première étape à réaliser lors de l'élaboration d'une solution à un problème tel que celui de la revue de code consiste en une étape de recherche. En effet, l'informatique s'étant fortement développée depuis quelques années, une multitude de programmes est maintenant à la portée de tous. Il est donc évident, afin de ne pas ré-inventer la roue, que la première étape consiste à vérifier s'il n'existe pas déjà un gestionnaire de revue de code qui pourrait être utilisé pour solutionner le problème de Club Capra.

Cependant, certains critères sont à prendre en compte. Premièrement, la solution ne doit pas être payante puisque nous travaillons pour des petites et très petites entreprises et leurs ressources sont donc limitées. Deuxièmement, la solution doit correspondre aux besoins réels de l'organisation : toutes les exigences de celle-ci doivent être remplies par la solution. Et finalement, et c'était le souhait des dirigeants de Club Capra, la solution devait rester simple, sous une licence libre et facile à utiliser tout en amenant une forte valeur ajoutée.

Sur base de ces différents critères, il n'a pas été permis de trouver une solution déjà existante. En effet, il existe peu de gestionnaires de revue de code. Et dans ceux existants, ils sont soit payants, soit non conformes aux besoins réels du club. Il a donc été décidé de créer un gestionnaire de revue de code : Code Review.

Pour le problème des tests unitaires, la méthode de tests unitaires JUnit (Java Unit testing) a été préférée. En effet, comme les membres de Club Capra travaillent principalement avec de la programmation java via le programme Eclipse, ces tests semblent être les plus appropriés. Ils permettent de vérifier qu'une fonction déterminée réalise bien la tâche qu'elle doit effectuer, et ce, via quelques lignes de code.

## 2.2 Description de la solution

### 2.2.1 Architecture du système

Le gestionnaire de revue de code développé est basé sur l'architecture de type "repository". Celle-ci se compose de deux types de modules. Le premier type est un repository centralisé où se trouvent les données partagées avec un modèle stable et un accès standard (Partie grisée de la figure 2.2.1) . Le deuxième est un ensemble de composants actifs quasi-autonomes bâtis autour du premier (Parties non grisées de la figure 2.2.1). C'est le système informatique classique avec une base de données centralisée. On peut alors représenter le gestionnaire par le schéma suivant :

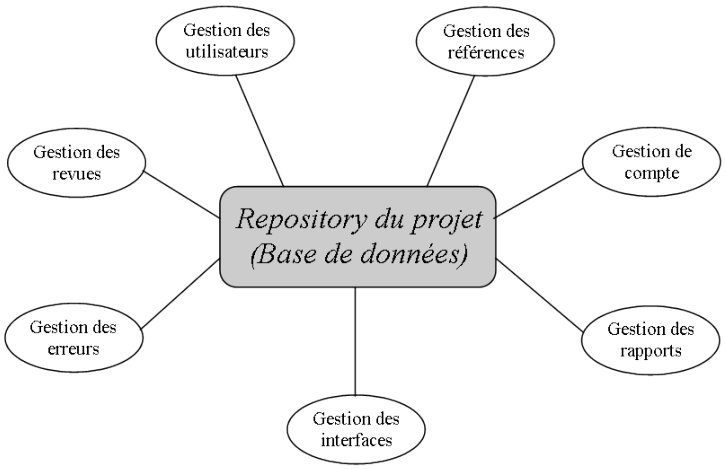



FIG. 2.1 – Architecture du gestionnaire de revue de code

Ce style d'architecture a été choisi pour différentes raisons telles que sa simplicité, son efficacité, son accès centralisé aux données permettant un haut niveau de sécurité, une bonne sauvegarde des données,... De plus, le repository ayant été installé sur un serveur, cela a permis de développer une application multi-utilisateurs où le serveur est joué par le repositiry tout en étant un module passif n'offrant qu'un accès centralisé aux données et où les clients sont joués par les différentes exécutions du gestionnaire de revue de code (chaque exécution se faisant sur un poste différent).

### 2.2.2 Présentation générale du système

Le programme "Code Review" a pour objectif principal l'apport d'un soutien informatique à la revue de code. En effet, celui-ci permet principalement l'enregistrement des erreurs détectées lors des phases de vérification et validation d'un projet. Afin d'être le plus efficace possible, certaines fonctionnalités ont été ajoutées telles qu'une gestion des utilisateurs, une gestion des références (fichiers) servant de base à la revue de code, une gestion d'exportation de rapport permettant d'extraire au format papier des informations contenues dans la base de données, ...

Dans un but d'homogénéité des sauvegardes d'erreurs, un template simple a été élaboré et ensuite transcrit en interface. Toutes les caractéristiques de celui-ci sont obligatoires lors de l'enregistrement d'une erreur. Voici ce template :

| |
|---|
| **Révision :** La phase de revue de code sous laquelle l'erreur est enregistrée. |
| **Responsable :** L'utilisateur qui enregistre l'erreur. |
| **Référence :** Le fichier dans lequel l'erreur a été détectée. |
| **Date de détection :** La date à laquelle l'erreur a été détectée. |
| **Date de correction :** La date à laquelle l'erreur devra être corrigée. |
| **Ligne :** Le numéro de ligne où l'erreur à été détectée. |
| **Description :** La description de l'erreur détectée. |
| **L'état :** Spécifie si l'erreur a déjà été corrigée. |

FIG. 2.2 – Template d'enregistrement d'une erreur

## 2.3 Cycle de développement du système

Afin de développer un système complet, fonctionnant correctement et répondant aux exigences du client, un cycle de vie en cascade avec une seule itération tel que représenté par la figure ci-dessous a été préféré. Ce choix d'une seule itération a été imposé par le peu de temps laissé au développement de la solution, à savoir 3 mois. En effet, avec plus de temps, travailler avec 2 itérations eut été possible : la première couvrant les fonctionnalités principales du système et la deuxième les secondaires.

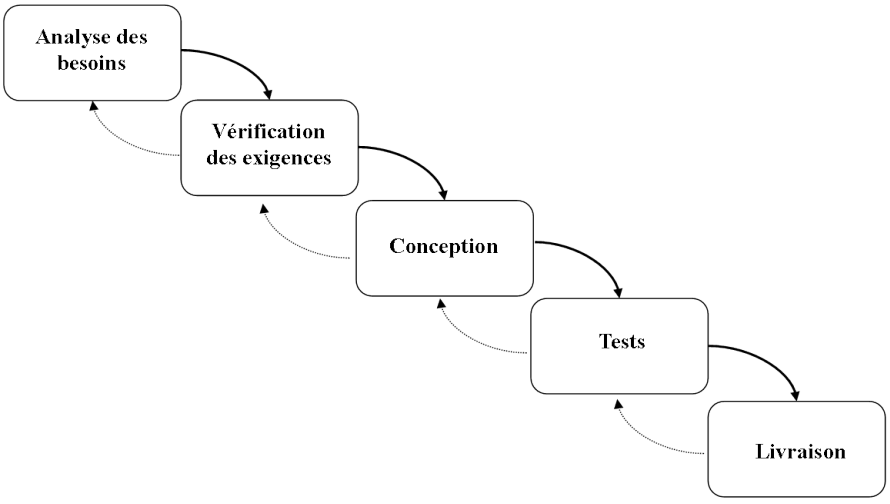



FIG. 2.3 – Cycle de développement du gestionnaire de revue de code

Les différentes phases qui se sont enchaînées pour aboutir au produit final peuvent être définies de la manière suivante :

1. **Analyse des besoins :** Son but est de modéliser les solutions techniques et organisationnelles susceptibles de répondre au problème posé. Lors de cette phase, on établit les diagrammes Use Cases, on décrit tout les scénarii possibles à partir de ces diagrammes et on schématise la statique des données du programme futur. Le résultat de cette phase est un cahier des charges reprenant toute l'analyse effectuée.

2. **Vérification des exigences :** Cette phase permet de vérifier que les exigences du client ont bien toutes été comprises par l'équipe d'implémentation mais aussi qu'elles sont toutes représentées dans le prototype. Cette phase consiste à réaliser des prototypes d'interface et à les valider ensuite avec le client. Aucune implémentation de fonctionnalités n'est entreprise. Le résultat de cette phase est le prototype final du système.

3. **Conception :** Elle consiste à implémenter le programme proprement dit sur base du prototype d'interface final et du cahier des charges établi lors de la phase d'analyse des besoins. Le résultat obtenu par cette phase est le programme proprement dit.

4. **Tests :** Son but est de vérifier et valider le programme. Un ensemble de jeux de test doit être établi et appliqué ensuite au programme afin de vérifier si celui-ci fonctionne correctement dans les bonnes situations et s'il est le plus robuste possible dans les situations critiques. Il est préférable de faire tester le programme par des personnes ne l'ayant pas implémentée. Une fois ces tests terminés, un programme complet et fiable est obtenu.

5. **Livraison :** Cette dernière phase consiste à livrer et installer le programme chez le client ainsi qu'à former celui-ci à son utilisation.

Les points importants relatifs à l'analyse des besoins réalisée pour ce gestionnaire de revue de code vont maintenant être passés en revue.

## 2.4 Analyse des besoins

### 2.4.1 Les classes d'utilisateurs

Le gestionnaire de revue de code offre des fonctionnalités différentes selon la classe à laquelle l'utilisateur appartient. Ces classes sont au nombre de deux :

#### L'administrateur

Tout administrateur doit être étudiant de l'ETS. Le but du travail de l'administrateur est de mettre à jour le système en y incluant les nouveaux utilisateurs. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel. Du fait de sa qualification et de son intérêt évident pour la revue de code, l'administrateur adopte une attitude positive vis-à-vis de la tâche. Sa langue maternelle est le français.

Les périphériques d'entrée utilisés sont le clavier et la souris. Le système d'exploitation de l'administrateur est Windows. Il n'y a pas de versions types de celui-ci (Professionnel ou Familial). Une machine virtuelle Java doit être installée. L'administrateur utilise le logiciel seul et principalement dans les locaux de Club Capra, à l'ETS, puisque l'accès à la base de données ne peut se faire que localement.

#### Le correcteur

Tout correcteur doit être étudiant de l'ETS. Le but du travail du correcteur est de répertorier dans le système les erreurs qu'il a pu trouver dans les fichiers lors de ses phases de révision de code ainsi que de tenir à jour les ajouts des nouvelles références. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel. Du fait de sa qualification et de son intérêt évident pour la revue de code, le correcteur adopte une attitude positive vis-à-vis de la tâche. Sa langue maternelle est le français.

Les périphériques d'entrée utilisés sont le clavier et la souris. Le système d'exploitation du correcteur est Windows. Il n'y a pas de versions types de celui-ci (Professionnel ou Familial). Une machine virtuelle Java doit être installée. Le correcteur utilise le logiciel seul et principalement dans les locaux de Club Capra, à l'ETS, puisque l'accès à la base de données ne peut se faire que localement.

### 2.4.2 Les fonctionnalités du système

Les Use Cases représentent le comportement affiché par le système sous certaines conditions, de manière à satisfaire un objectif de l'un des acteurs. Les Use Case Diagrams montrent quant à eux les acteurs, les limites du système, les relations entre les acteurs et le système ainsi que les relations entre les Use Cases eux-mêmes. Ceux-ci permettent, aux travers de scénarii, de capturer, de représenter et de valider les fonctions d'un domaine, de les spécifier et de donner un aperçu de la dynamique et des interactions. Du fait de leur simplicité, les Use Cases ont pour avantage notable d'être compréhensibles par tous.

Les fonctionnalités principales du système peuvent être divisées en deux parties : celles attribuées à l'administrateur et celles au correcteur. Il est à noter que l'administrateur peut aussi remplir les fonctionnalités du correcteur.

#### Du côté de l'administrateur

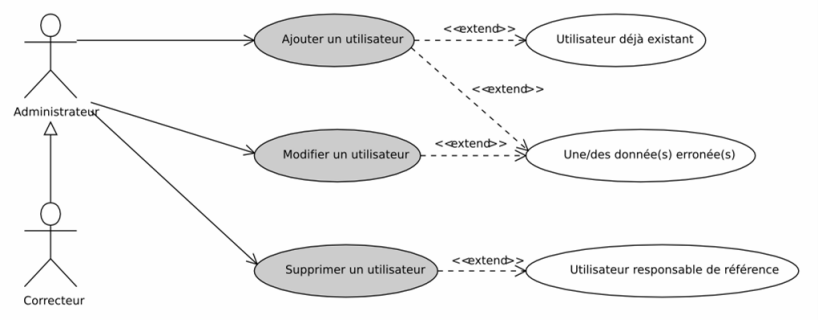



Fig. 2.4 – Diagramme Use Cases de l'administrateur

Les fonctionnalités offertes à l'administrateur ont trait à la gestion des utilisateurs et peuvent se définir de la manière suivante :

1. *Ajouter un utilisateur :* Permet à l'administrateur de rentrer un nouvel utilisateur dans le système. Pour cela, il doit spécifier certaines informations telles que son nom, son prénom, sa fonction, son statut... Un login numérique unique est alors attribué au nouvel utilisateur.

2. *Modifier un utilisateur :* Permet à l'administrateur de modifier un utilisateur enregistré dans le système. Seules les informations relatives à la fonction et au statut sont modifiables.

3. *Supprimer un utilisateur :* Permet à l'administrateur de supprimer un utilisateur enregistré dans le système.

### Du côté du correcteur

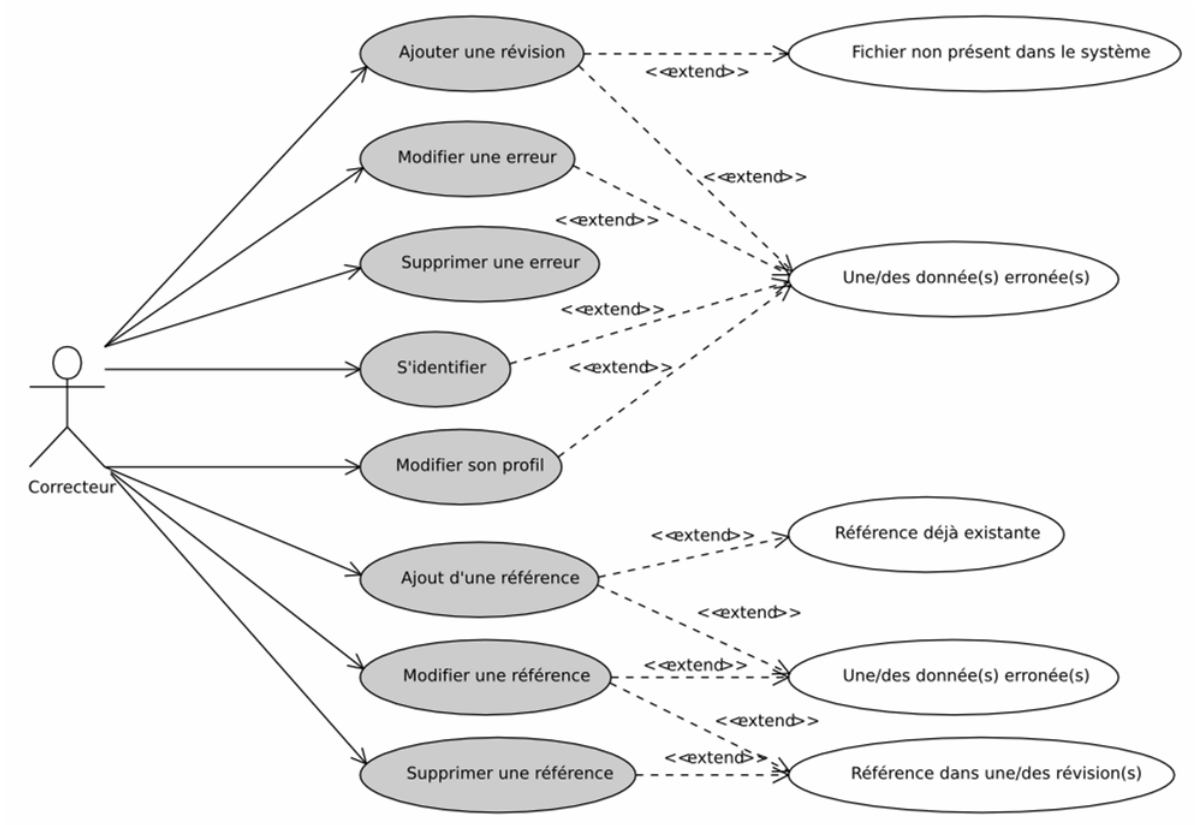



Fig. 2.5 – Diagramme Use Cases du correcteur

Les fonctionnalités offertes au correcteur ont trait à la gestion des phases de révision de code et aux références, et peuvent se définir de la manière suivante :

1. *Ajouter une révision :* Permet au correcteur d'ajouter un ensemble d'erreurs à une nouvelle phase révision de code ou à une phase déjà existante dans le système. Cette fonctionnalité est basée sur le template de la figure 2.2.

2. *Modifier une erreur :* Permet au correcteur de modifier certaines informations relatives à une erreur contenue dans une phase de révision de code enregistrée dans le système. Il peut ainsi spécifier si une erreur a été corrigée ou non.

3. *Supprimer une erreur :* Permet au correcteur de supprimer une erreur, enregistrée dans le système, qui n'a plus d'utilité pour le projet auquel elle est associée.

4. *S'identifier :* Permet au correcteur de s'identifier au système en spécifiant son login et son mot de passe qui lui ont été attribués par un administrateur lors de la création de son compte.

5. *Modifier son profil :* Permet au correcteur de modifier certaines informations de son profil telles que son mot de passe, son adresse e-mail, ...

6. *Ajouter une référence :* Permet au correcteur d'ajouter une/des nouvelle(s) référence(s) dans le système.

7. *Modifier une référence :* Permet au correcteur de modifier l'emplacement d'une référence dans le cas où celle-ci aurait été déplacée.

8. *Supprimer une référence :* Permet au correcteur de supprimer une référence du système dans le cas où celle-ci aurait été supprimée du projet.

### 2.4.3 Représentation de la statique des données

Le programme réalisé pour Club Capra se base sur une base de données MySql installée sur leur serveur. Celle-ci peut être représentée via un modèle Entité-Relation-Attribut. Ce modèle propose des concepts, principalement les entités, les associations et les attributs, permettant de décrire un ensemble de données relatives à un domaine défini. On obtient alors le schéma suivant :

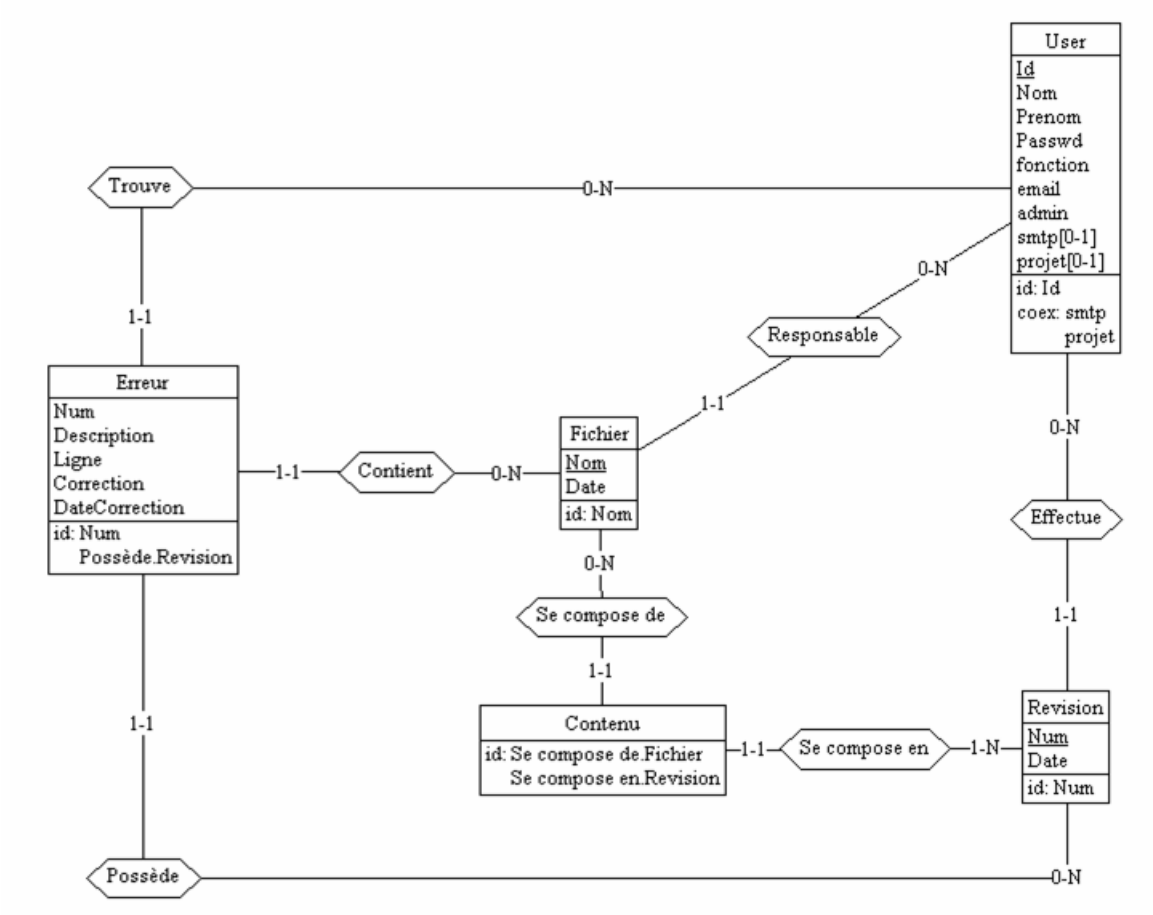



FIG. 2.6 – Schéma de la statique des données

Les différentes tables présentes dans la base de données peuvent être décrites brièvement de la manière suivante :

*Table User*

Cette table contient les utilisateurs enregistrés dans le système. Chaque utilisateur est identifié par un login numérique à valeur unique qui est généré automatiquement. L'attribut *"admin"* permet de classer l'utilisateur en tant qu'administrateur ou en tant que correcteur.

*Table Fichier*

Cette table contient l'ensemble des références enregistrées dans le système et sur lesquelles les phases de révision de code peuvent être réalisées. Chaque référence est identifiée par un nom générique représentant le nom de la référence précédé de ses répertoires parents.

*Table Revision*

Cette table contient les phases de révision de code enregistrées dans le système. Chaque phase est identifiée par un numéro de révision unique généré automatiquement lors de l'enregistrement de chaque phase.

*Table Erreur*

Cette dernière table contient l'ensemble des erreurs enregistrées lors des phases de révision de code par un utilisateur. Chaque erreur est identifiée par un numéro d'erreur qui est une valeur numérique unique générée automatiquement et par un numéro de révision contenu dans le table "Revision".

## 2.5 Conclusion

Sur base de ce cahier des charges, l'implémentation et la vérification de la solution permettant d'améliorer la gestion de la qualité chez Club Capra ont pu être entreprises. Cette phase a cependant été placée en annexe puisque celle-ci fait référence à un domaine plus technique. Une fois la solution implémentée et avant de passer à la phase de livraison et formation, une méthodologie devait aussi être proposée, afin de mener à bien la révision de code. Cette méthodologie est présentée dans la section suivante.

# Section 3 :

## *Méthodologie proposée*

> **Contenu :** *Présentation de la méthodologie proposée à Club Capra pour réaliser leur révision de code.*

En programmation, les tests unitaires sont un procédé permettant de s'assurer du fonctionnement correct d'une partie déterminée d'un logiciel ou d'une portion d'un programme. Il s'agit pour le programmeur de tester un module, indépendamment du reste du programme, ceci afin de s'assurer qu'il répond aux spécifications fonctionnelles et qu'il fonctionne correctement en toutes circonstances (Adaptée de [21]).

La méthodologie proposée à Club Capra se base sur les tests unitaires JUnit. Le choix de cette méthode repose sur différentes raisons. Premièrement, les dirigeants du club ont spécifié leur intention de travailler avec ce type de test pour leurs projets futurs et deuxièmement, comme leur projet se fait principalement en programmation java, les tests unitaires JUnit semblaient être les plus appropriés.

Deux exemples illustrent cette méthode : le premier se base sur une simple fonction d'addition basique tandis que le deuxième se base sur une fonction tirée du code du programme réalisé pour Club Capra.

### 3.0.1 Exemple 1

La fonction utilisée pour illustrer ce premier exemple réalise l'addition de deux nombres passés en argument et retourne ensuite le résultat. Le code de la fonction est le suivant :

```
public static int add(int a, int b) {
     return a + b;
}
```

Pour tester cette fonction via un test unitaire JUnit, il est nécessaire de mettre dans la classe de test la fonction ci-dessous qui, via la commande *assertTrue()* et à partir des arguments (2,2) va tester si la fonction *add* renvoie bien l'entier 4. Dans ce cas, l'exécution du test mentionnera que la fonction est correctement codée puisque 2 + 2 = 4.

```
public void testAdd() {
     assertTrue(fonctionBasic.add(2,2) == 4);
}
```

Supposons maintenant qu'une erreur est commise dans l'implémentation de la fonction *add* tel que celle-ci se définit par le bout de code suivant :

```
public static int add(int a, int b) {
     return a - b;
}
```

Si l'on test la fonction *add* avec la fonction de test établie plus haut, l'exécution de la fonction *testAdd* indiquera que la fonction testée ne retourne pas l'élément spécifié puisque le résultat sera 0 et non 4. La fonction *add* est donc à vérifier.

### 3.0.2 Exemple 2

Le deuxième exemple se base sur une fonction dont la tâche est de vérifier que la string passée en argument contient bien un entier. Dans ce cas, la fonction retourne la valeur *true*, sinon *false*. Le code de la fonction est le suivant :

```
public static boolean isInteger(String s){
     try{
          new Integer(s);
          return true;
     }
     catch(NumberFormatException e){
          return false;
     }
}
```

On peut alors tester la fonction *isInteger* via la fonction de test ci-dessous. La première instruction va tester si la fonction *isInteger* avec un argument équivalent à "1" renvoie bien **true**. La deuxième va tester si la fonction *isInteger* avec un argument équivalent à "a" renvoie bien **false**. Dans les deux cas, l'exécution du test se fera correctement, déclarant ainsi que la fonction *isInteger* réalise bien la tâche demandée.

```
public void testIsInteger() {
     assertTrue(fonctionNonBasic.isInteger("1") == true);
     assertTrue(fonctionNonBasic.isInteger("a") == false);
}
```

# Section 4 :

## *Résultats du 1$^e$cycle d'amélioration*

> **Contenu :** *Description de la livraison de la solution ainsi que des différents problèmes rencontrés durant l'intervention. Elle se termine par une conclusion générale sur la démarche effectuée.*

### 4.1 Livraison du produit

Une fois le programme implémenté et la méthodologie élaborée, la solution a pu être livrée à Club Capra. Cette livraison s'est déroulée en deux phases : l'installation du gestionnaire et la formation au gestionnaire et à la méthodologie l'accompagnant. Ces deux phases se sont déroulées dans les locaux du club et n'ont rencontré aucun problème.

### 4.2 Problèmes rencontrés

Aucun problème majeur durant notre intervention chez Club Capra n'a été rencontré. En fait, suite à l'expérience acquise lors des différents projets réalisés durant nos candidatures et maîtrises en informatique à l'Institut d'Informatique de Namur (Belgique), les aspects techniques tels que la programmation en java, la gestion d'une base de données ou la conception d'un cahier des charges et d'implémentation ne nous ont pas posé beaucoup de problèmes.

Cependant, au point de vue relation avec les dirigeants de l'organisation, une difficulté de suivi de ceux-ci durant la conception et l'implémentation de la solution peut être soulignée. En effet, à plusieurs moments, nous avons dû prendre contact à maintes reprises afin de recevoir une réponse à notre question. Cependant, étant des étudiants, nous comprenons que les dirigeants de Club Capra ne pouvaient pas toujours répondre dans les plus brefs délais. Mais, suite à leur motivation lors de la première entrevue, nous espérions plus de motivation de leur part.

## 4.3 Résultats du premier cycle d'amélioration

Les réelles motivations des dirigeants de Club Capra pour entamer une amélioration de leurs processus logiciels étaientt tout d'abord de profiter de ressources extérieures au club. Ainsi, certains outils pourraient être développés afin de soutenir leurs processus futurs. En effet, étant fortement occupés par les différentes compétitions se déroulant au cours de l'année, les ressources nécessaires pour développer de tels outils ne sont pas disponibles au sein du club pour le moment.

A ce jour, les membres de Club Capra ne rencontrent pas de problèmes quant à l'utilisation de l'application et méthodologie. De plus, ceux-ci veulent continuer à faire évoluer le gestionnaire en y intégrant par exemple un plugin permettant de faire le lien direct entre le répertoire SVN, sur lequel leur projet est stocké, et le gestionnaire.

Nous pouvons donc conclure en mentionnant que les dirigeants sont très satisfaits de notre solution mais aussi, du temps qu'ils nous ont consacré afin que nous les aidions à améliorer la qualité de leur processus logiciel actuel.

## 4.4 Cycles d'amélioration futurs

Le travail que nous avons réalisé pour Club Capra ne représente que le premier cycle d'amélioration de leurs processus et pratiques logiciels. Ainsi, d'autres pratiques telles que celles liées à leur méthodologie de développement ou de gestion des changements pourraient être la cible du prochain cycle d'amélioration. Cependant, si le temps attribué au prochain cycle le permet, nous suggérons d'améliorer les pratiques relatives à leur méthodologie de développement puisque cette pratique nous semble importante. De plus, son amélioration amènera une grande valeur ajoutée dans l'organisation. Dans le cas où le temps attribué est trop court, nous suggérons d'améliorer la gestion des changements et ce, pour les mêmes raisons.

## Troisième partie

# Deuxième étude de cas

# Section 1 :

## *Sélection de l'amélioration*

> **Contenu :** *Cette section reprend l'identification et l'analyse des différentes pratiques à améliorer chez Horizon Vert ainsi que la sélection d'un de celles-ci. Elle commence par présenter l'entreprise.*

### 1.1 Présentation de l'entreprise

La deuxième entreprise étudiée fut l'entreprise d'Horizon Vert Centre-du-Québec[1] située à Nicolet, Québec, Canada. Cette PME travaille dans le secteur agricole et se compose de 9 personnes dont Monsieur Jean-François Ménard qui en est le président depuis 1997.

Horizon Vert consacre environ 60% de ses activités dans la gestion d'un regroupement de producteurs agricoles à but non-lucratif dans l'agro-environnement et environ 40% de ses activités dans la conception web avec leur projet SAAP. Ce projet est un outil efficace permettant de gérer ses terres agricoles via son ordinateur[2].

### 1.2 Problèmes présents dans l'entreprise

La première étape consiste à élaborer les différents problèmes relevés par le rapport de la micro-évaluation [2] réalisé sur Horizon Vert afin d'en établir un plan d'action. Ce rapport est réalisé dans le cadre du projet d'APPEQ[3] et permet d'établir les forces, les faiblesses, les opportunités et les risques, ainsi que les recommandations à court et long terme relatives à l'entreprise évaluée.

[1] Appelée simplement Horizon Vert dans la suite du mémoire.
[2] Pour de plus amples informations sur ce projet, veuillez consulter la section 2.5 de cette partie.
[3] **A**mélioration de la **P**erformance des **P**etites **E**ntrerprises **Q**uébecoises

Le tableau synoptique de la figure 1.2 représente la situation d'Horizon Vert selon les différents axes de la micro-évaluation. On peut constater que sur certains axes tels que la formation ou la gestion des versions, Horizon Vert est dans une bonne situation alors que pour les axes tels que la gestion des exigences et des changements, l'organisation est dans une situation inverse.

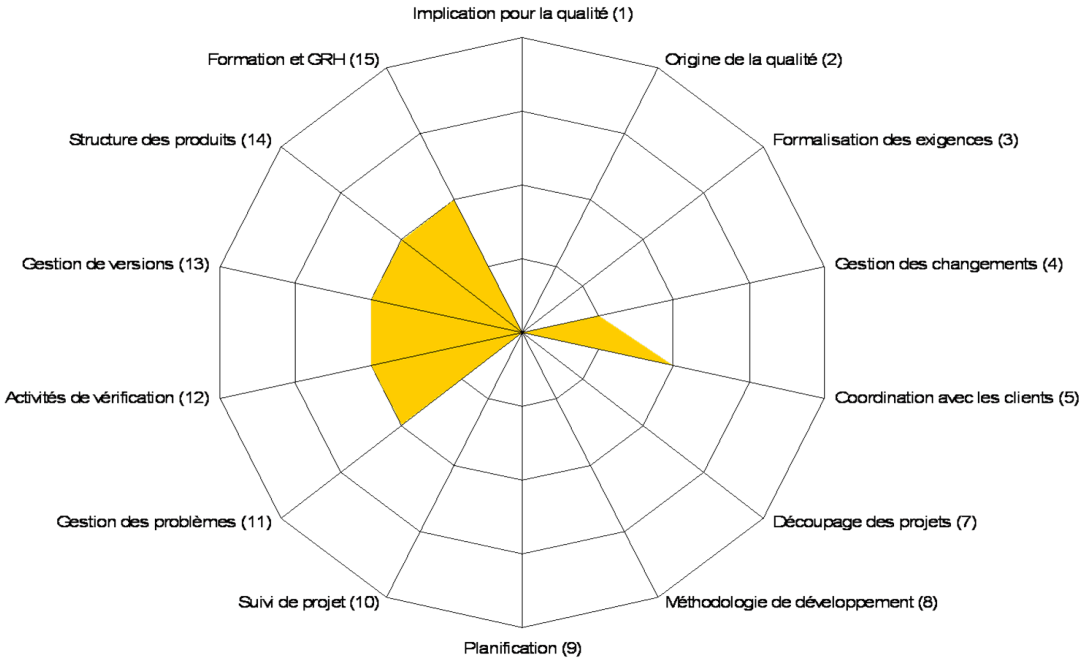



Fig. 1.1 – Synoptique détaillée d'Horizon Vert

Les différentes pratiques à améliorer identifiées au sein de l'entreprise grâce au rapport de la micro-évaluation vont être énoncées ci-dessous [2] :

### Gestion des exigences

Le premier problème relevé chez Horizon Vert concerne la gestion de ses exigences, c'est-à-dire la formalisation et la spécification de celles-ci. En effet, l'entreprise ne travaille pas avec un processus de formalisation et de spécification de ses exigences. Quant aux changements pouvant les influencer, aucun processus n'est déclenché. En effet, le client ne transmet ses changements que par un simple coup de fil. Ceux-ci sont ensuite notés sur un bout de papier.

### Gestion de la documentation

Le deuxième problème se réfère à la gestion de la documentation. Horizon Vert ne possède pas de gabarit pour structurer les documents d'analyse qu'elle réalise et il en est de même pour les livrables. Une partie de ce problème pourrait être solutionné par une bonne gestion des exigences et des changements.

### Gestion de projet

Le troisième problème concerne la gestion de projet. Au sein de l'entreprise Horizon vert, aucune gestion formelle de projet ni d'estimation formelle pour les projets d'envergure n'ont été mises en place.

### Méthodologie de développement

En dernier point se trouve la méthodologie de développement. Horizon Vert ne possède pas de cycle de vie formalisé ni de méthodologie de développement de logiciel. Ce problème se réfère à du moyen terme.

## 1.3 Critères de sélection de la pratique à améliorer

Une fois les différentes pratiques à améliorer répertoriées, il ne reste plus qu'à établir les critères de choix permettant d'éliminer celles qui ne peuvent être solutionnées. La partie pratique du stage ne durant que 3 mois, le premier critère retenu fut celui du court terme. En effet, il n'était pas possible d'améliorer des pratiques relatives à du moyen terme (plus de 6 mois). Il semble donc évident que la pratique relative à la méthodologie de développement ne soit retenue.

L'objectif de l'intervention est d'améliorer une pratique logicielle de l'organisation via une solution peu coûteuse et qui amène une forte valeur ajoutée. Il est donc logique que celle liée à la gestion de projet ne soit pas retenue puisque celle-ci est déjà présente en partie dans l'entreprise.

Pour les deux restantes, la gestion de la documentation peut être mise sur le côté puisque l'amélioration de cette pratique peut être résolue en partie via une bonne gestion des exigences et des changements.

## 1.4 Choix de la pratique à améliorer

Grâce à l'énumération des différentes pratiques à améliorer sur base du rapport de la micro-évaluation ainsi que l'élaboration des critères de sélection, une proposition d'amélioration de leur pratique relative à la gestion des exigences et ensuite celle relative à la gestion des changements d'exigences a été faite aux dirigeants d'Horizon Vert. Après une entrevue avec les dirigeants de l'entreprise, cette proposition a été acceptée et l'élaboration et l'implémentation ont pu commencer. Ces améliorations seront donc les deux premiers cycles d'amélioration des processus et pratiques logiciels chez Horizon Vert.

# Section 2 :

## *Premier cycle d'amélioration*

> **Contenu :** *Description et élaboration de la solution d'amélioration de la pratique relative à la gestion des exigences c'est-à-dire la phase d'analyse nécessaire à sa bonne implémentation.*

### 2.1 Recherche d'une solution

La première étape à réaliser lors de l'élaboration d'une solution à un problème tel que celui de la spécification et formalisation des exigences consiste en une étape de recherche. En effet, l'informatique s'étant fortement développée depuis quelques années, une multitude de programmes est maintenant à la portée de tous. Il est donc évident, afin de ne pas ré-inventer la roue, que la première étape consiste à vérifier s'il n'existe pas déjà un solution à ce problème.

Certains critères sont à prendre en compte lors de cette recherche. Ceux-ci sont comparables à ceux utilisés pour la recherche d'une solution pour Club Capra. Sur base de ces différents critères, une solution déjà existante a pu être sélectionnée : le programme GenSpec. Ce programme est développé par l'entreprise Hydro-Québec, important distributeur, transporteur et producteur d'électricité en Amérique du Nord siégeant à Montréal.

### 2.2 Pourquoi choisir GenSpec ?

L'outil GenSpec est un gestionnaire d'exigences complet, facile à utiliser et en évolution continue. Son utilisation est gratuite et entièrement en français. De plus, il offre un panel d'avantages si on le compare à un logiciel de traitement de texte tel que Word ou OpenOffice Writer. Quelques exemples peuvent être cités tels que (repris de [3] et [4]) :

1. **Réduction des coûts :** Grâce à GenSpec, les coûts de l'ingénierie des exigences sont fortement réduits puisque les efforts sont concentrés sur les exigences plutôt que sur leur présentation, les rapports d'évaluation sont générés automatiquement, GenSpec génère automatiquement des textes de début d'exigence.

2. **Facilité de lecture :** GenSpec offre une bonne structure des exigences allant de la vue la plus générale à la plus détaillée ainsi que des options pour la génération des documents.

3. **Réduction de la quantité d'erreurs :** GenSpec offre un vérificateur d'exigence, gère un historique de modification des exigences, empêche d'introduire des incohérence de hiérarchie.

4. **Respect de normes internationales :** GenSpec fixe un numéro unique par exigence, fait abstraction des moyens de réalisation, définit la priorité de chacune des exigences.

## 2.3 Présentation de GenSpec

GenSpec propose 4 fonctionnalités principales et 4 fonctionnalités secondaires via une interface divisée en 2 parties : la partie de droite représentant l'arbre hiérarchique des exigences du projet et la partie de gauche, la description de l'exigence sélectionnée préalablement dans l'arbre. En voici la description des fonctionnalités principales (Description adaptée de [3] , [4]).

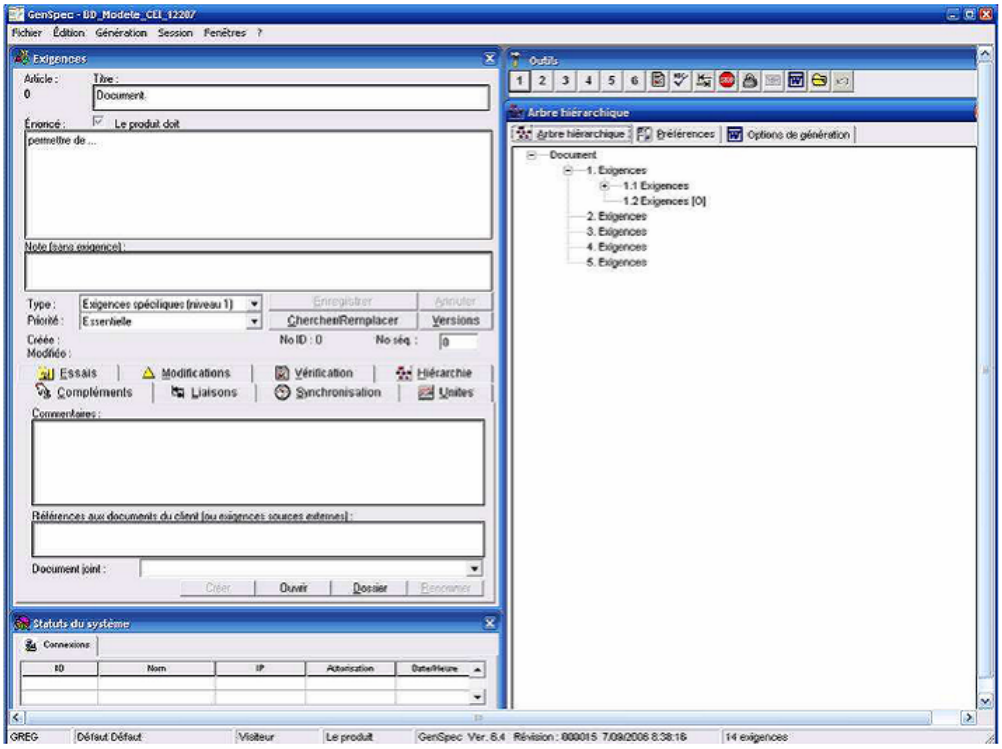



FIG. 2.1 – Présentation de GenSepc

### Structure des exigences

La première fonctionnalité offerte par GenSpec est une structuration claire et simple des exigences du projet via un arbre hiérarchique de 6 niveaux. Une exigence du niveau N représente une généralisation des exigences filles. Plus on se rapproche du dernier niveau de l'arbre, plus les exigences deviennent spécifiques.

### Définition des exigences

Une fois l'arbre hiérarchique construit, chaque de ses exigences peut être définie via différentes caractéristiques telles que :

- le titre qui équivaut au nom donné à l'exigence dans l'arbre,
- l'énoncé c'est-à-dire une description de l'exigence,
- le type qui équivaut à celui indiqué lors de l'ajout dans l'arbre : Exigence spécifique - Exigence de haut niveau - Macro-fonction - Fonction - Détail[1] et,
- la priorité de réalisation : Essentielle - Optionnelle - Obligatoire.

### Exportation des exigences

La troisième fonctionnalité principale de GenSpec est son exportation de document. En effet, une fois votre projet en développement ou terminé, vous pouvez exporter son contenu grâce à différents types de documents d'exportation : les exigences et leurs caractéristiques, l'arbre hiérarchique, l'historique de modification,... De plus, certaines options d'exportation peuvent être définies afin de générer un document clair et ne contenant que les éléments dont vous avez besoin.

### Vérification des exigences

La dernière fonctionnalité repose dans la vérification des exigences. Afin de vérifier qu'il n'y a aucune erreur dans la structure et la description du projet, un vérificateur d'exigence est disponible. Celui-ci inspectera chaque exigence de l'arbre et spécifiera par exemple que certaines ne possèdent pas d'énoncé ou que certaines sont d'un type incohérent par rapport au type de leur exigence parente.

En plus de ses fonctionnalités principales, GenSpec offre une gestion des utilisateurs permettant d'ajouter, modifier et supprimer des utilisateurs du système, un accès sécurisé au projet via une identification obligatoire lors de son ouverture et d'autres fonctionnalités relatives à la description des exigences telles que garder un historique des changements subis par l'exigence depuis son insertion dans le programme ou revenir à des versions antérieures du projet.

## 2.4 Méthodologie d'utilisation

Après une phase d'apprentissage et de test sur GenSpec, une méthodologie simple a pu être élaborée afin d'aider les employés d'Horizon Vert dans l'utilisation future de ce programme. Cette méthodologie se divise en deux phases : premièrement, la structuration de l'arbre et deuxièmement, la définition des exigences présentes dans l'arbre.

[1] D'autres types sont présents dans GenSpec mais n'ont pas été retenus lors de notre étude, ceux-ci s'avérant non nécessaire pour Horizon Vert.

### 2.4.1 Structuration de l'arbre

Lors de l'ajout d'une nouvelle exigence dans l'arbre hiérarchique, il est nécessaire de définir son type. Celui-ci dépend fortement du niveau auquel l'exigence sera placée ainsi que de l'exigence qui en sera la parente. L'arbre hiérarchique peut donc être construit de la manière suivante (du niveau le plus abstrait au plus concret) :

*Niveau 1*

Ce niveau ne peut comporter qu'un seul type d'exigences : les exigences spécifiques. C'est un terme global pour définir toute exigence de la spécification. Le nom donné par défaut à cette exigence doit rester inchangé.

*Niveau 2 :*

Ce niveau ne contient que les exigences de haut niveau, à savoir les exigences fonctionnelles, les contraintes de conceptions, les exigences de performance,... Les noms donnés par défaut à ces exigences doivent rester inchangés.

*Niveau 3 :*

Le troisième niveau ne contient que des exigences de type macro-fonction. Ce sont en fait des "modules exigences" qui regroupent un ensemble de fonctionnalités ayant trait à la même partie d'un programme. Le nom donné à une macro-fonction doit être du type "Gestion de...". Par exemple, avoir une macro-fonction "Gestion des utilisateurs" ou "Gestion des révisions de code".

*Niveau 4 :*

Le quatrième niveau peut contenir deux types d'exigences : un macro-fonction s'il y a nécessité de définir le module d'exigence du niveau supérieur en sous-modules ou une fonction qui présente une fonctionnalité future du système. Dans le premier cas, le module "Gestion des utilisateurs" du niveau 3 pourrait avoir été divisé en deux sous-modules "Gestion des administrateurs" et "Gestion des utilisateurs lambda".

Dans le deuxième cas, des fonctionnalités pourraient être définies telles qu'"Ajouter un utilisateur" ou "Supprimer un utilisateur" sous le module "Gestion des utilisateurs" de niveau 3.

*Niveau 5 :*

Tout comme le niveau précédent, ce niveau peut aussi contenir deux types d'exigences : une fonction si l'exigence parente est une macro-fonction ou un détail si l'exigence parente est une fonction. Un détail est une magnification d'une exigence, un niveau de définition supplémentaire.

Par exemple, si au niveau 4 existe une fonction "Ajouter un utilisateur", pourraient alors être spécifiés au niveau 5 des détails tels que nom, prénom, adresse,...

*Niveau 6 :*

Ce dernier niveau ne peut contenir qu'un seul type d'exigence : les détails. Il ne sera utilisé que si le niveau 5 contient des fonctions et que l'on désire les définir plus précisément.

Via les différents exemples cités ci-dessus, un arbre hiérarchique peut être établi. La figure de gauche représente l'arbre dans le cas où le module "Gestion des utilisateurs" a été divisé en deux sous-modules tandis que la figure de droite représente l'arbre dans le cas où celui-ci n'a pas été divisé en sous-modules mais directement en fonctions.

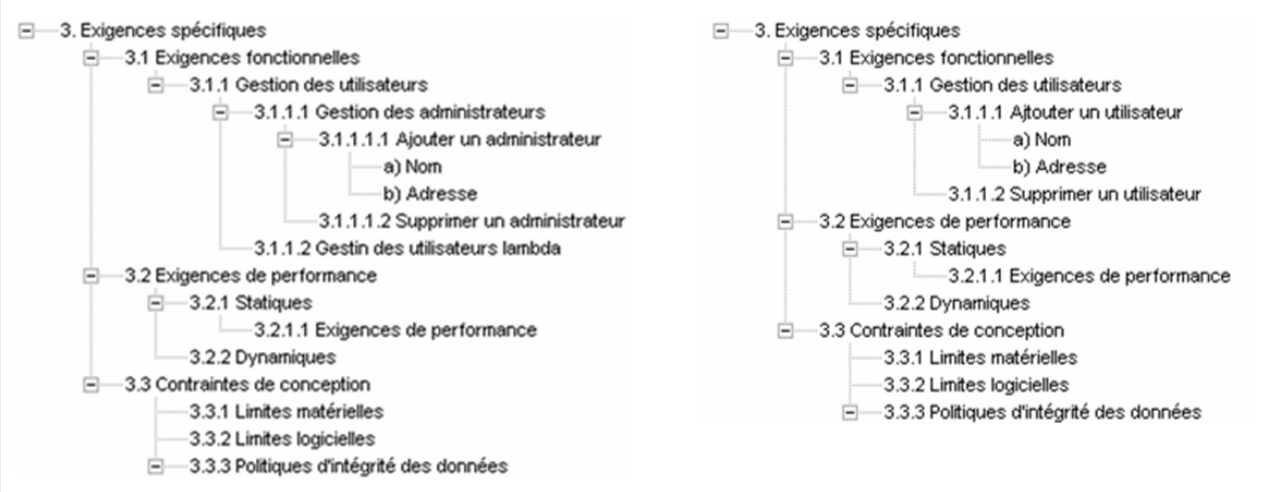



Fig. 2.2 – Exemples de structure d'arbre

### 2.4.2 Définition des exigences

Une fois l'arbre hiérarchique construit, il reste à définir l'énoncé et la priorité de l'exigence qui est par défaut "Essentielle"[2]. Voici les caractéristiques une à une :

*Énoncé*

L'énoncé décrit ce que doit réaliser l'exigence s'il est lié à une exigence ne représentant pas un détail de fonction. De ce cas, il est impératif d'inclure dans cette description ce qu'est l'exigence, comment la réaliser et qui est susceptible de la réaliser (quoi - comment - qui). Dans le cadre d'un détail, l'énoncé représente la description du détail.

[2]Le titre de l'exigence et le type ayant été spécifiés lors de la construction de l'arbre

Exemple : la fonction "ajouter un utilisateur". Un énoncé possible pourrait être : *Le projet doit permettre à un utilisateur possédant des droits d'administrateur d'ajouter un nouvel utilisateur dans le système en remplissant les champs donnés en détail de la fonctionnalité.* On retrouve dans cette description les trois éléments essentiels :

- **Quoi :** Permettre d'ajouter un nouvel utilisateur dans le système.
- **Comment :** En remplissant un certain nombre de champs.
- **Qui :** Un utilisateur possédant des droits d'administrateur.

*Priorité*

Le degré de nécessité doit être choisi entre trois valeurs différentes : soit Essentielle si l'exigence est obligatoire, Complémentaire si l'exigence n'est qu'un complément au projet (pour des fonctionnalités secondaires par exemple) ou Optionnelle si l'exigence ne sera développée que si le temps ou les ressources le permettent.

## 2.5 Cas d'utilisation

Une fois la prise en main de GenSpec effectuée ainsi qu'une méthodologie d'utilisation implémentée, ce programme a été testé sur base d'un projet en cours d'Horizon Vert : l'outil SAAP. C'est l'outil de gestion web idéal pour aider à gérer des champs : consultation des champs, modification des cultures et mesure des distances sans bouger de son fauteuil. L'outil SAAP permet aussi de gérer la fertilisation, les semences, les analyses de sols, les importations, les animaux. De plus, il est sécuritaire et confidentiel : il suffit d'un mot de passe et d'un accès haute vitesse pour y accéder (Description adaptée de [20]).

Avant de commencer ce projet, Horizon Vert avait réalisé une petite phase d'analyse des exigences et établi quelques cas d'utilisation. Sur base de ces documents, et sans aucune connaissance du projet, il a été tenté de spécifier les exigences de celui-ci via GenSpec. Le travail s'est avéré plus que difficile car ces cas d'utilisation étaient simples. Mais grâce à cette expérience, le vrai potentiel de GenSepc a pu être démontré et aussi l'objectif d'une bonne spécification des exigences, ce qui a satisfait les dirigeants d'Horizon Vert.

# Section 3 :

## *Deuxième cycle d'amélioration*

> **Contenu :** *Description de l'élaboration de la solution d'amélioration de la pratique relative à la gestion des changements c'est-à-dire la phase d'analyse nécessaire à sa bonne implémentation.*

## 3.1 Recherche d'une solution

Après avoir entrepris une phase de recherche identique à celle nécessaire pour solutionner le problème de gestion des exigences d'Horizon Vert, et sur base des mêmes critères de recherche, aucune solution existante au problème de gestion des changements d'exigences n'a pu être trouvée. C'est pourquoi, un gestionnaire de requêtes de changement a été implémenté : modeX.

## 3.2 Description de la solution

### 3.2.1 Architecture du système

Le gestionnaire de requêtes de changement développé pour Horizon Vert est basé sur le même style d'architecture que celui utilisé pour le gestionnaire de revue de code destiné à Club Capra : un repository centralisé où se trouvent les données partagées avec un modèle stable et un accès standard (Partie grisée de la figure 3.2.1) et un ensemble de composants actifs quasi-autonomes bâtis autour de ce repository (Parties non grisée de la figure 3.2.1). L'architecture du gestionnaire peut être représenté par la figure 3.2.1 ci-après.

Ce style d'architecture a été choisi pour les mêmes raisons que celles citées pour Club Capra. De plus, le repository ayant été installé sur un serveur, cela a permis de développer une application multi-utilisateurs où le serveur est joué par le repository tout en étant un module passif n'offrant qu'un accès centralisé aux données et où les clients sont joués par les différentes exécutions du gestionnaire de requêtes de changement (chaque exécution se faisant sur un poste différent).

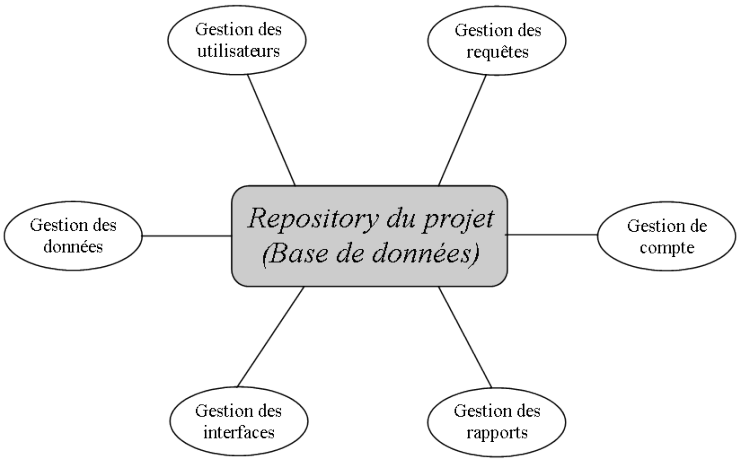



FIG. 3.1 – Architecture du gestionnaire de requêtes de changement

### 3.2.2 Présentation générale du système

Le programme développé pour Horizon Vert a pour objectif principal la gestion des requêtes de changement d'exigences pouvant survenir à tout moment lors de la réalisation d'un projet. Par requête de changement, il faut entendre l'ajout, la modification ou la suppression d'une exigence. La fonctionnalité principale du système réside donc dans le gestion de ces requêtes qui est basée sur un processus et un template décris ci-après.

Afin d'être le plus efficace possible, certaines fonctionnalités ont été ajoutées telles qu'une gestion des utilisateurs, une gestion d'exportation de rapport permettant d'extraire au format papier des informations contenues dans la base de données...

## 3.3 Processus et Template

### 3.3.1 Processus de manipulation d'une requête

Le processus de gestion des changements, représenté à la figure 3.3.1, a pour but d'aider l'entreprise à gérer formellement les requêtes de changement provenant de leurs clients.

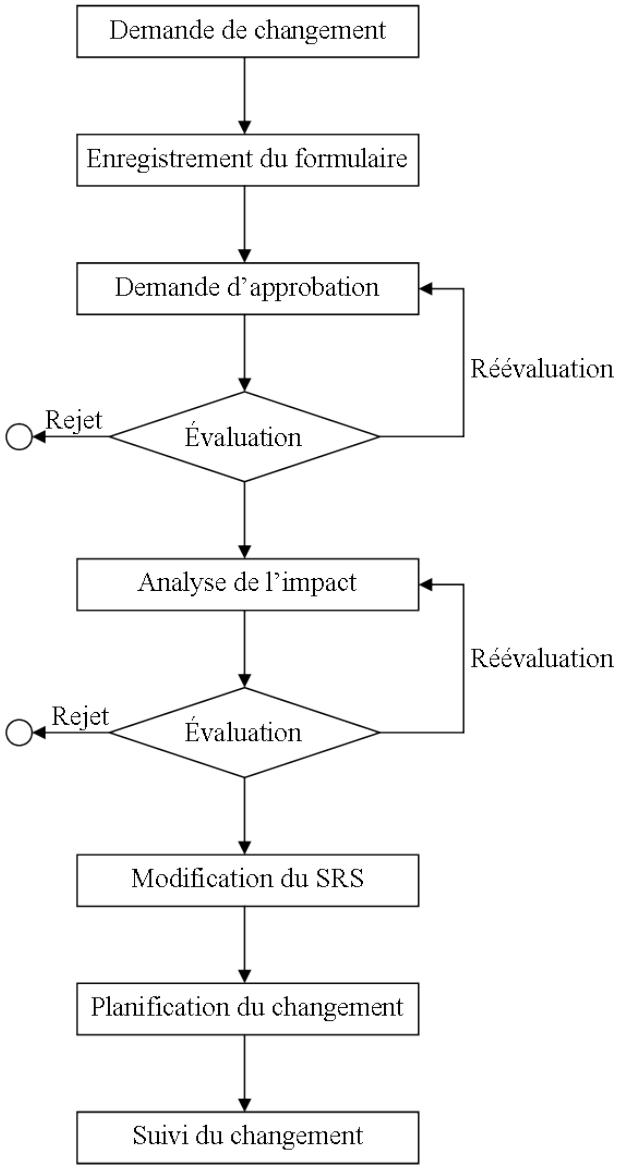



FIG. 3.2 – Processus de gestion des changements

Chaque étape du processus correspond à un état spécifique de la requête de changement. Les différentes étapes ainsi que l'état qui lui correspond sont ici décrites.

*Demande de changement*

Un nouveau processus prend place à chaque nouvel demande de changement de la part du client. Pour ce faire, le client contacte l'entreprise et lui transmet sa demande. Les différentes possibilités de contact sont le téléphone, les e-mails, ...

*Enregistrement du formulaire*

Cette étape représente l'enregistrement de la requête de changement via un formulaire bien défini et basé sur le template défini en 3.3.2. Ce template permet d'obtenir une description détaillée et homogène des requêtes de changement. À la fin de cette étape, le statut de la requête de changement est à "enregistrée".

*Demande de validation*

Cette première évaluation va permettre de décider s'il y a lieu ou non de réaliser la requête de changement. C'est le chef de projet qui est responsable d'évaluer la faisabilité du changement et de contacter éventuellement le client pour de plus amples informations. À la fin de cette étape, le statut de la requête de changement est à "validée" ou "rejetée".

*Analyse de l'impact*

Cette seconde évaluation va permettre de décider s'il y a lieu ou non de réaliser la requête de changement, préalablement validée, en fonction de son impact sur le système existant. Le chef de projet va donc étudier l'impact que le changement aurait sur les autres exigences et prendre en considération toutes les alternatives possibles à ce changement. À la fin de cette étape, le statut de la requête de changement est à "approuvée" ou "rejetée".

*Modification du SRS*

À cette étape, il s'agit de traduire la requête de changement, préalablement approuvée, via l'outil GenSpec. Un changement dans le système peut impliquer soit, une nouvelle exigence pour le système, soit une modification d'une ou plusieurs exigences existantes. À la fin de cette étape, le statut de la requête de changement est à "spécifiée".

*Planification du changement*

Une fois le changement spécifié, il s'agit de planifier sa réalisation. Le chef de projet est donc responsable de répartir, dans le temps, les tâches nécessaires à la réalisation du changement entre les membres de son équipe. À la fin de cette étape, le statut de la requête de changement est à "planifiée".

*Réalisation du changement*

Lors de cette étape, les ressources allouées au développement du changement vont réaliser ce dernier en respectant la planification préétablie. À la fin de cette étape, le statut de la requête de changement est à "réalisée".

*Suivi du changement*

Cette dernière étape consiste en un suivi de la réalisation du changement qui a été planifié. Pour ce faire, le chef de projet consulte régulièrement les ressources allouées afin de connaître l'avancement du développement et constitue, ensuite, une équipe de testeurs qui se chargera de réaliser des plans de tests bien définis afin de vérifier la modification apportée au système. À la fin de cette étape, le statut de la requête de changement est à "testée".

### 3.3.2 Template d'enregistrement d'une requête

Cette partie décrit le modèle de formulaire utilisé pour l'enregistrement des requêtes de changement provenant des clients. Ce modèle est composé de trois parties complémentaires, à savoir, l'*identification*, la *description* et l'*analyse*[1].

Les parties concernant l'*identification* et la *description* de la demande seront enregistrées par la personne en charge du point de contact avec le client. Les trois parties pourront ensuite être mises à jour par le chef de projet, au fur et à mesure de la progression de la demande à travers les étapes du processus.

#### Identification

Cette première partie du template concerne l'identification du changement d'exigences et contient l'ensemble des caractéristiques d'identification de la requête de changement.

| |
|---|
| **Nom du projet* :** Le nom du produit à développer. |
| **Numéro de la requête* :** Le numéro de la requête dans le projet. |
| **Titre* :** Le titre résumant le but de la requête. |
| **Émise par* :** Le client qui émet la requête. |
| **Enregistrée par* :** Le point de contact du client, celui qui enregistre la requête. |
| **Priorité* :** Un niveau de priorité de la requête basé sur trois niveaux. |
| **État* :** L'état de la requête, tel que précisé dans le processus de gestion des changements. |
| **Date de demande* :** La date à laquelle la requête a été émise par le client. |
| **Date d'échéance :** La date à laquelle la requête devient inutile pour le client. |

[1]Les champs munis d'un * sont définis comme obligatoire lors de l'enregistrement de la requête de changement

### Description

Après l'identification de la requête, on passe ensuite à la description de celle-ci, ce qui va permettre d'obtenir plus d'informations sur la requête de changement.

| |
|---|
| **Type* :** Le type de la requête, tels que : Ajout, Modification, Retrait,... |
| **Description* :** Une description de la requête, de ce que le client voudrait. |
| **Description originale :** Une description de l'existant. |
| **Justification* :** La raison qui justifie la réalisation de la requête. |
| **Commentaire technique :** Un commentaire technique sur la requête. |
| **Documents relatifs :** Des documents annexes permettant d'illustrer la requête. |

### Analyse

Cette dernière partie du template ne peut être remplie que par le chef du projet et contient les caractéristiques relatives aux phases d'évaluation et de planification du processus préalablement présenté.

*Évaluation*

| |
|---|
| **Aperçu de l'impact* :** Description de l'impact du changement sur les autres exigences. |
| **Exigences concernées :** La liste des numéros d'exigences qui sont concernées par le changement demandé. |
| **Alternatives :** Les alternatives possibles relatives au changement demandé. |
| **Action* :** Ce qu'il convient d'entreprendre pour réaliser le changement. |

*Planification*

| |
|---|
| **Analyste* :** La personne responsable d'évaluer et de planifier le changement. |
| **Approbateur* :** La personne qui donne son accord pour réaliser le changement. |
| **Implémenteur* :** La personne responsable d'implémenter le changement. |
| **Date d'approbation :** La date à laquelle l'approbateur a donné son accord. |
| **Date de début des travaux :** La date planifiée de début des travaux. |
| **Date de fin des travaux :** La date planifiée de fin des travaux. |
| **Date finale :** La date de réalisation finale du changement. |

## 3.4 Cycle de développement du système

Le gestionnaire de requête de changement a été développé sur base du même cycle de vie que celui utilisé pour la réalisation du gestionnaire de revue de code de Club Capra, et ce pour les mêmes raisons.

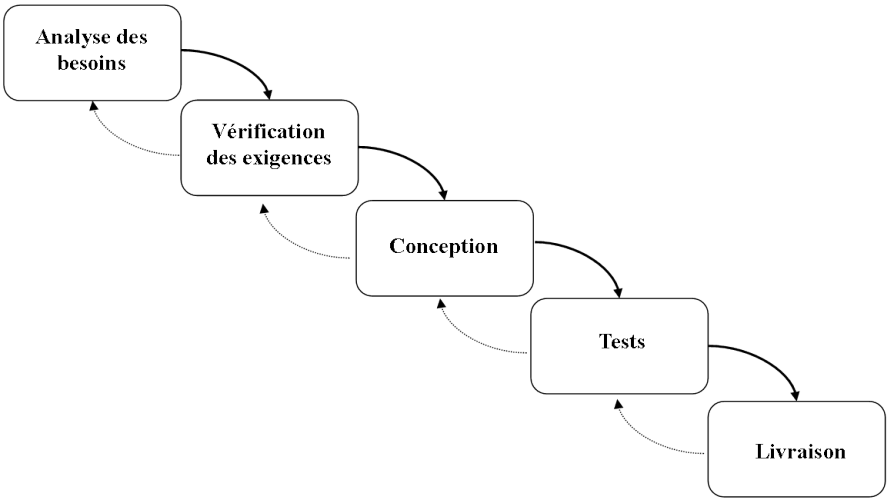



Fig. 3.3 – Cycle de développement du gestionnaire de requête de changement

Voici les points importants relatifs à l'analyse des besoins réalisée pour modeX.

## 3.5 Analyse des besoins

### 3.5.1 Les classes d'utilisateurs

Le gestionnaire de changement de requête offre des fonctionnalités différentes selon la classe à laquelle l'utilisateur appartient. Ces classes sont au nombre de deux :

#### Chef de projet

Le but du travail du chef de projet est de mettre à jour le système en gérant les demandes de changement ainsi qu'en gérant l'ajout et la suppression d'utilisateurs. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel. Du fait de sa qualification et de son intérêt évident pour la gestion des changements, le chef de projet adopte une attitude positive vis-à-vis de la tâche. Sa langue maternelle est le français.

Les périphériques d'entrée utilisés sont le clavier et la souris. Le système d'exploitation du chef de projet est Windows XP. Il n'y a pas de versions types de celui-ci (Professionnel ou Familial). Une machine virtuelle Java doit être installée. Le chef de projet utilise le logiciel seul et principalement dans les locaux de l'entreprise puisque l'accès à la base de données ne peut se faire que localement.

### Utilisateur

Le but du travail de l'utilisateur est de mettre à jour le système en y incluant les nouvelles demandes de changement. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel. Du fait de sa qualification et de son intérêt évident pour la gestion des changements, l'utilisateur adopte une attitude positive vis-à-vis de la tâche. Sa langue maternelle est le français.

Les périphériques d'entrée utilisés sont le clavier et la souris. Le système d'exploitation du chef de projet est Windows XP. Il n'y a pas de versions types de celui-ci (Professionnel ou Familial). Une machine virtuelle Java doit être installée. De même que le chef de projet, il utilise le logiciel seul et principalement dans les locaux de l'entreprise puisque l'accès à la base de données ne peut se faire que localement.

## 3.5.2 Les fonctionnalités du système

Les Use Cases représentent le comportement affiché par le système sous certaines conditions de manière à satisfaire un objectif de l'un des acteurs. Les Use Case Diagrams montrent quant à eux les acteurs, les limites du système, les relations entre les acteurs et le système ainsi que les relations entre les Use Cases eux-mêmes. Ceux-ci permettent, aux travers de scénarii, de capturer, de représenter et de valider les fonctions d'un domaine, de les spécifier et de donner un aperçu de la dynamique et des interactions. Du fait de leur simplicité, les Use Cases ont pour avantage notable d'être compréhensibles par tous.

Les fonctionnalités principales du système peuvent être divisées en deux parties : celles attribuées au chef de projet et celles à l'utilisateur. Il est à noter que le chef de projet peut aussi remplir les fonctionnalités de l'utilisateur.

### Du côté du chef de projet

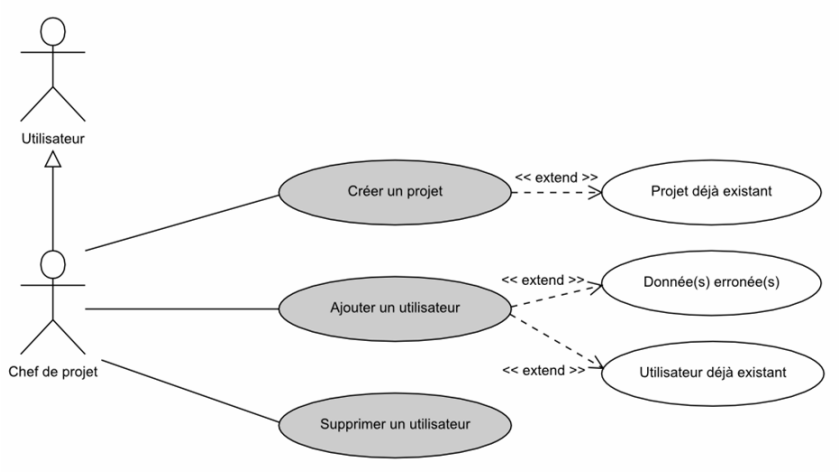



Fig. 3.4 – Diagramme Use Cases du chef de projet

Les fonctionnalités offertes au chef de projet ont trait à la gestion des utilisateurs et la gestion de la partie "analyse" des requêtes de changement. Elles peuvent se définir de la manière suivante :

1. *Créer un projet :* Permet au chef de projet d'ajouter un nouveau projet dans le système.

2. *Ajouter un utilisateur :* Permet au chef de projet de rentrer un nouvel utilisateur dans le système. Pour cela, il doit spécifier certaines informations telles que son nom, son prénom, son statut... Un login numérique unique est alors attribué au nouvel utilisateur.

3. *Supprimer un utilisateur :* Permet au chef de projet de supprimer un utilisateur enregistré dans le système.

### Du côté de l'utilisateur

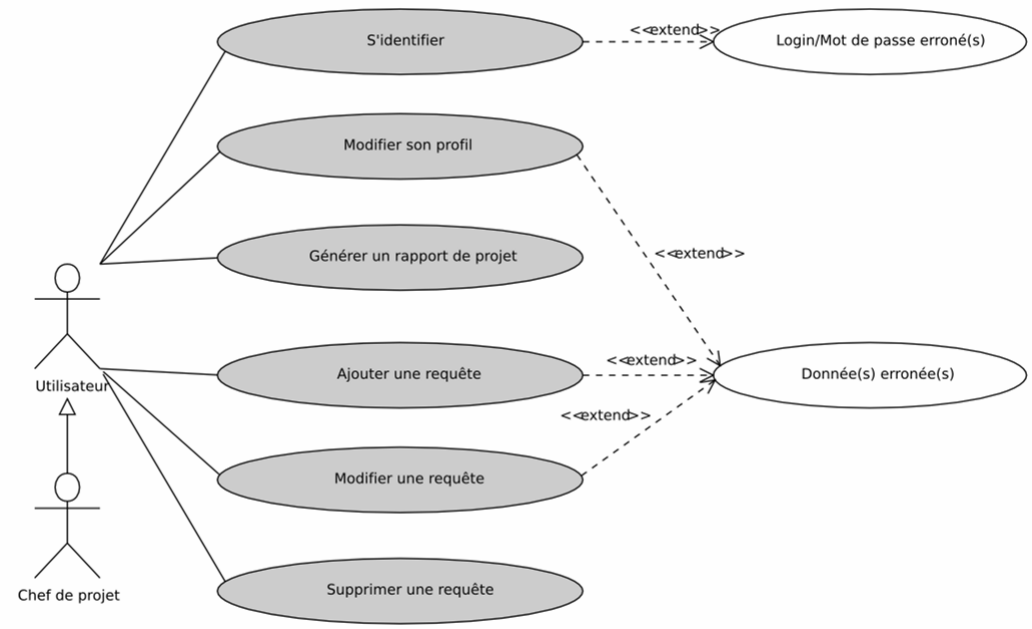



Fig. 3.5 – Diagramme Use Cases de l'utilisateur

Les fonctionnalités offertes par le gestionnaire à l'utilisateur ont trait à la gestion des requêtes de changement et peuvent se définir de la manière suivante :

1. *Ajouter une requête :* Permet à l'utilisateur d'ajouter une nouvelle requête de changement dans le système sur base du template défini précédemment. Seule la troisième partie du template (voir figure 3.3.2) peut être remplie par le chef de projet.

2. *Modifier une requête :* Permet à l'utilisateur de modifier une requête de changement préalablement enregistrée dans le système, exceptée la partie Analyse qui est réservée au Chef de projet.

3. *Supprimer une requête :* Permet à l'utilisateur de supprimer une requête de changement préalablement enregistrée dans le système.

4. *Générer un rapport de projet :* Permet à l'utilisateur de générer le rapport d'un projet existant dans le système en y incluant les phases du template qu'il désire.

5. *S'identifier :* Permet à l'utilisateur de s'identifier au système en spécifiant son login et son mot de passe qui lui ont été attribués par le chef de projet lors de la création de son compte.

6. *Modifier son profil :* Permet à l'utilisateur de modifier certaines informations de son profil telles que son mot de passe, son adresse e-mail,...

### 3.5.3 Représentation de la statique des données

Le programme réalisé pour Horizon Vert se base sur une base de données MySql installée sur leur serveur. Celle-ci peut être représentée via un modèle Entité-Relation-Attribut. Ce modèle propose des concepts, principalement les entités, les associations et les attributs, permettant de décrire un ensemble de données relatives à un domaine défini. On obtient alors le schéma suivant :

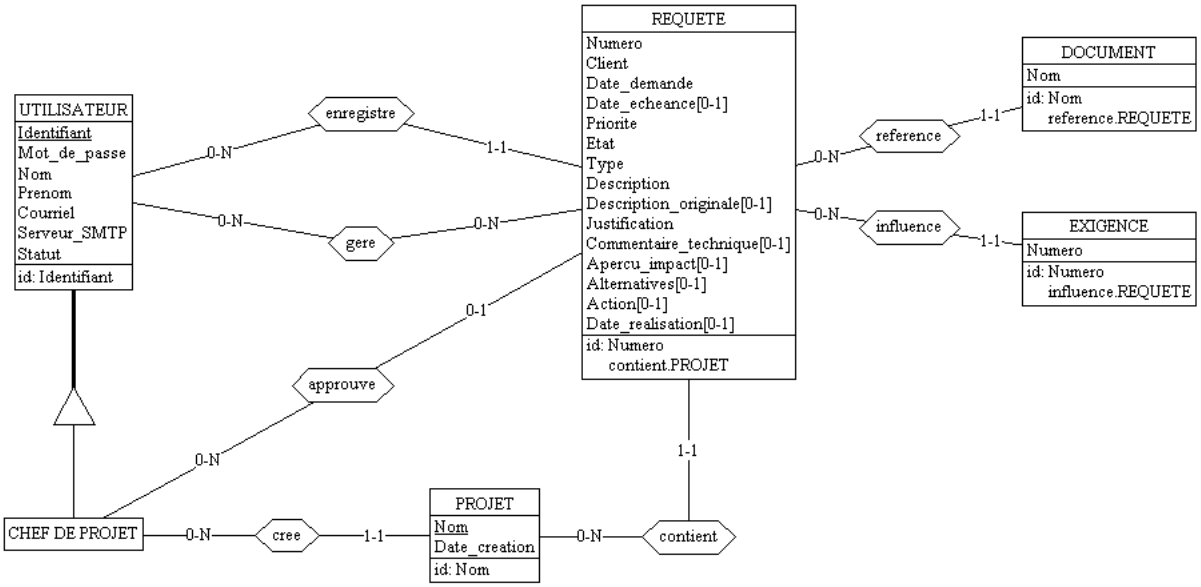



FIG. 3.6 – Schéma de la statique des données

Les différentes tables présentes dans la base de données peuvent être décrites brièvement de la manière suivante :

*Utilisateur*

Cette table contient les utilisateurs enregistrés dans le système. Chaque utilisateur est identifié par un login numérique à valeur unique qui est généré automatiquement.

*Projet*

Cette table contient l'ensemble des projets enregistrés dans le système. Chaque projet est identifié par un nom unique.

*Requete*

Cette table contient l'ensemble des requêtes de changement enregistrées dans le système via le formulaire d'enregistrement. Les champs munis d'une cardinalité 0-1 correspondent aux champs non obligatoires du template (voir figure 3.3.2). Chaque requête est identifiée par un numéro unique au sein du projet auquel elle se rapporte.

*Exigence*

Cette table contient toutes les exigences contenues dans GenSpec qui sont influencées par au moins une requête de changement enregistrée dans le système. Chaque exigence est identifiée par un numéro unique et par les identifiants de la requête y faisant référence.

*Document*

Cette table contient tous les documents (fichiers externes au programme) qui sont référencés par au moins une requête de changement enregistrée dans le système. Chaque document est identifié par un nom unique et par les identifiants de la requête y faisant référence.

## 3.6 Conclusion

Sur base de ce cahier des charges, l'implémentation et la vérification de la solution permettant d'améliorer la gestion des changements d'exigences chez Horizon Vert ont pu être entreprises. Cette phase a cependant été placée en annexe puisque celle-ci fait référence à un domaine plus technique. Une fois la solution implémentée, nous sommes passés à l'installation de notre solution et à la formation à celle-ci.

# Section 4 :

## *Résultats des 2 cycles d'amélioration*

> **Contenu :** *Description de la livraison des solutions élaborées ainsi que des différents problèmes rencontrés durant l'intervention. Elle se termine par une conclusion générale sur Horizon Vert.*

### 4.1 Livraison du produit

Une fois les solutions implémentées, elles ont pu être livrées à Horizon Vert. Cette livraison s'est déroulée en deux temps : premièrement le programme GenSpec et puis le gestionnaire modeX. En effet, GenSpec étant prêt avant de commencer modeX, il a été préféré de le livrer directement afin de terminer complètement le premier cycle d'amélioration qui lui était associé. Cette première livraison s'est déroulée dans les locaux de l'entreprise, à Nicolet, et n'a rencontrée aucun problème.

En ce qui concerne le second cycle d'amélioration, celui relatif au problème de gestion des changements des exigences, la même phase de livraison que celle réalisée pour GenSpec n'a pu être réitérée. En effet, les locaux de l'entreprise ne se situant pas à proximité de Montréal, il était difficile de s'y rendre une seconde fois. Cependant, le prototype d'interface ayant déjà été démontré et comme peu de changements devaient y être apportés, une formation n'était plus vraiment nécessaire. Le gestionnaire a donc été livré à Horizon Vert par le biais d'Internet. Après installation et quelques essais, les dirigeants d'Horizon Vert ont reporté n'avoir rencontré aucun problème.

### 4.2 Problèmes rencontrés

Aucun problème majeur durant notre intervention chez Horizon Vert n'a été rencontré. En fait, suite à l'expérience acquise lors de l'élaboration de la solution au problème de revue de code de Club Capra, les aspects techniques tels que la programmation en java, la gestion d'une base de données ou la conception d'un cahier des charges et d'implémentation ne nous ont pas posé de problèmes.

Cependant, comme spécifié dans notre première étude de cas, une difficulté de suivi des dirigeants de l'entreprise durant la conception et l'implémentation de nos deux solutions peut être soulignée. En effet, étant une petite entreprise québécoise, ils leur était difficile de nous consacrer beaucoup de temps. Cependant, nous avons pu profiter pleinement du temps qu'ils ont pu nous consacrer.

## 4.3 Résultats du premier cycle d'amélioration

Suite au départ du poste d'analyse de monsieur Bobby Richard et à l'importance de cette phase de développement qu'il a apportée au sein de l'entreprise, les dirigeants d'Horizon Vert voulaient continuer dans cette même direction. C'est pourquoi, ils ont lancé cette phase d'amélioration de la qualité de leurs processus et pratiques logiciels.

En ce qui concerne GenSpec, le programme est peu utilisé pour le moment. Cela est dû au fait que la personne qui va remplacer monsieur Richard a déjà des modules à programmer et ne peut donc y consacrer tout son temps. Les dirigeants d'Horizon Vert en sont cependant très satisfaits, ainsi que du temps passé pour arriver à ce résultat. Il en est de même pour notre méthodologie d'utilisation.

## 4.4 Résultats du deuxième cycle d'amélioration

Au point de vue de modeX, nous pouvons tirer les mêmes conclusions et constatations que celles réalisées sur GenSpec puisqu'ils sont tout à fait satisfaits du programme et de la méthodologie que nous leur avons livrés.

## 4.5 Cycles d'amélioration futurs

Le travail que nous avons réalisé pour Horizon Vert ne représente que les deux premiers cycles d'amélioration de leurs processus et pratiques logiciels. Ainsi, d'autres pratiques telles que celles liées à leur suivi des modules ou aux phases de vérifications de projets peuvent être la cible du prochain cycle d'amélioration. Ces deux pratiques ayant été suggérées par les dirigeants d'Horizon-Vert, il nous semble logique que le choix d'une des deux se face directement avec ceux-ci.

## Quatrième partie

# Critiques de la version québécoise de la micro-évaluation

# Section 1 :

## *Améliorations possibles*

> **Contenu :** *Présentation des améliorations pouvant être utiles d'apporter à la version québécoise de la micro-évaluation.*

## 1.1 Introduction

Sur base du stage et de l'expérience acquise durant celui-ci, il est maintenant possible de prendre un certain recul face aux méthodes et outils utilisés afin de les améliorer. En effet, plusieurs problèmes ont été rencontrés durant l'utilisation de la micro-évaluation. Ces commentaires ont permis de faire ressortir certaines améliorations. Ce sont ces améliorations qui vont maintenant être présentées.

Notons que dans la suite de cette section, les éléments encadrés représentent les améliorations pouvant être apportées à la micro-évaluation.

## 1.2 Exactitude des questions

La première amélioration qui peut être apportée à la micro-évaluation concerne l'exactitude de ses questions. En effet, certaines questions contenues dans le questionnaire sont vagues. Le problème est que ce type de question peut amener des réponses vagues, rendant ainsi la réponse non exploitable puisque trop interprétable. Ce problème devient plus important si le représentant de l'organisation évaluée a peu d'expérience. L'ensemble du questionnaire va maintenant être parcouru tout en proposant les améliorations possibles pour chaque axe.

### Gestion de la qualité

Le premier point du questionnaire pouvant faire l'objet d'une amélioration concerne la gestion de la qualité (A) (voir 2.4.1 de la partie 1). Le problème est que si l'organisation évaluée ne possède pas de gestion de qualité, la seule réponse qu'elle fournira aux différentes questions ci-dessus est qu'elle ne possède pas de gestion de qualité. Il pourrait alors être intéressant de questionner le représentant de l'entreprise sur la raison pour laquelle son entreprise n'en possède pas. On pourrait alors opter pour une structure comme celle-ci :

1. Est-ce que votre entreprise a déjà entamé des activités qualités ?
   - *Si oui,*
     - De quel type ?
     - Qu'est ce que cela vous a apporté ?
     - Quels sont les moyens mis en oeuvre pour garantir cette qualité ?
   - *Si non,*
     - Pour quelles raisons ?
2. En résumé, au niveau "Gestion de la qualité (A)", estimez-vous que ce qui est fait est efficace et permet d'atteindre les résultats attendus ?

### Relation clients

En ce qui concerne la partie "Relation clients", la question "Formalisez-vous les exigences de vos clients" reste assez vague. En effet, il serait peut-être utile d'éclaircir cette question au moyen de diverses sous-questions telles que les outils utilisés pour formaliser les exigences ou encore les documents générés une fois la formalisation terminée. On obtiendrait alors une structure telle que celle ci-dessous. Les autres questions de cette partie restant inchangées.

1. Formalisez-vous les exigences de vos clients ?
   - Quels outils utilisez-vous pour formaliser les exigences de vos clients (Use case,...) ?
   - Quels documents, rapports générez-vous une fois cette étape terminée ?
2. Comment les clients vous font-ils part des modifications à apporter aux fonctionnalités attendues ?
   - Y a-t-il une trace écrite des demandes de modifications ?
   - Vos clients utilisent-ils un formulaire ou une procédure pour les demandes de modifications ?
3. Organisez-vous des réunions de coordination avec vos clients en cours de projet ?
   - S'agit-il de réunions régulières ou organisées à la demande ?
4. En résumé, au niveau "Relation clients (B)", estimez-vous que ce qui est fait est efficace et permet d'atteindre les résultats attendus ?

### Suite du questionnaire

Les autres parties du questionnaire doivent quant à elles rester inchangées. En effet, pour chaque question, il est possible de l'affiner grâce à ses diverses sous-questions.

## 1.3 Définition du client

Un des points traités par la micro-évaluation concerne les relations que l'organisation entretient avec ses différents clients. Le problème est que le questionnaire ne définit pas entièrement le terme "client". En effet, l'ensemble des questions et sous-questions, telles que reprises ci-dessous, laissent sous-entendre qu'un client est une personne externe à l'organisation et qui est demandeur d'un produit ou service à cette organisation.

1. Formalisez-vous les exigences de vos clients ?
   - Quels outils utilisez-vous pour formaliser les exigences de vos clients (Use case,...) ?
   - Quels documents, rapports générez-vous une fois cette étape terminée ?
2. Comment les clients vous font-ils part des modifications à apporter aux fonctionnalités attendues ?
   - Y a-t-il une trace écrite des demandes de modifications ?
   - Vos clients utilisent-ils un formulaire ou une procédure pour les demandes de modifications ?
3. Organisez-vous des réunions de coordination avec vos clients en cours de projet ?
   - S'agit-il de réunions régulières ou organisées à la demande ?

Le problème est qu'une organisation ne développe pas toujours des produits pour des clients qui lui sont externes. En effet, il est possible qu'elle développe un produit pour elle-même. Dans ce cas, le client doit être joué par l'employé de l'entreprise pour qui le produit est destiné. Or cette interprétation du mot client n'est pas implicite dans les différentes questions y faisant référence. Ainsi, il est possible de retrouver des rapports de micro-évaluation ne contenant aucune description sur les relations que l'organisation entretient avec ses clients.

Une solution possible est d'apporter dans la rubrique "Relations Clients" une définition de ce qu'est un client. Ainsi, on pourrait définir le terme client comme suit :

> **Un client** est un acteur qui demande la fourniture d'un produit ou d'un service à une organisation, qu'il soit externe ou interne à celle-ci. Tout projet contient donc un client.

## 1.4 Termes techniques

La micro-évaluation veut être perçue comme une méthode utilisant un vocabulaire simple, compréhensible par tous et composée par le moins possible de termes techniques. Cependant, certains termes utilisés dans le questionnaire peuvent être non ou mal compris pour le représentant de l'organisation évaluée. Ce problème de vocabulaire peut avoir un impact important sur la réponse que va donner le représentant. Prenons les questions suivantes comme exemple :

1. Formalisez-vous les exigences de vos clients ? (**A. Gestion de la qualité**)
2. Suivez-vous une méthodologie de développement pour vos projets ? (**D. Développement et gestion de projet**)

Il peut alors être difficile pour une personne ayant peu ou aucune expérience dans le domaine informatique de comprendre parfaitement ces deux questions. En effet, les termes tels que "formaliser" ou "méthodologie de développement" ne sont pas des termes de la vie courante. La réponse fournie pourrait alors être erronée suite à une mauvaise compréhension de la question.

Afin de résoudre ce problème de vocabulaire, la solution que nous proposons est l'utilisation d'un lexique reprenant les différents termes pouvant poser des problèmes de compréhension. Étant trop qualifiés pour mettre en évidence les termes trop techniques, nous avons demandé à différentes personnes, n'ayant pas de connaissances en informatique, de lire le questionnaire de la micro-évaluation et de nous spécifier les termes non compris. Après avoir regroupé les différents résultats obtenus, le lexique fut le suivant[1] :

1. **Activités qualités :** Activités permettant d'accroître la qualité d'un produit ou d'un service c'est-à-dire accroître l'aptitude d'un ensemble de caractéristiques intrinsèques à satisfaire des exigences (sécurité, ergonomie, performance,...).
2. **Cahier des charges :** Cahier visant à décrire exhaustivement les spécifications de base d'un produit ou d'un service à réaliser. Il sert à formaliser les besoins du client et à les expliquer aux différents acteurs pour s'assurer que tout le monde est d'accord sur la façon de procéder.
3. **Canevas :** Ébauche de document schématisant les lignes principales du document à réaliser.
4. **Cycle de vie :** Ensemble ordonné de phases décrivant la vie d'un projet, la phase n ne pouvant commencer que si la phase n-1 est terminée.
5. **Formaliser :** Consiste à exprimer techniquement les exigences du client via des outils tels que les diagrammes Use Cases.
6. **Livrable :** Tout composant matérialisant le résultat de la prestation de réalisation, c'est à dire toute production émise par le titulaire au cours du projet : document, courrier, module de code logiciel, dossiers de tests, application intégrée...
7. **Méthodologie de développement :** Procédure à appliquer pas à pas afin d'entreprendre l'ensemble des étapes et processus qui permettent de passer de l'expression d'un besoin informatique à un logiciel fonctionnel et stable.
8. **Outil de gestion de configuration :** Outil utilisé afin de stocker et tracer les différentes versions ou révisions de toute information destinée à être utilisée par un système (matériel, logiciel, document, données,...).
9. **Template :** Modèle de conception logiciel ou de représentation des données.

[1] L'ensemble des définitions reprises dans le lexique provient de [21] et [27]

## 1.5 Présentation de l'organisation évaluée

La quatrième amélioration pouvant être proposée pour la micro-évaluation concerne la présentation de l'organisation évaluée. En effet, il peut être intéressant pour la personne en charge du processus d'amélioration de l'organisation évaluée de mieux cibler, de mieux comprendre l'organisation pour laquelle elle va travailler. S'imprégner du contexte de l'organisation permet d'élaborer des solutions plus adaptées à celle-ci. Par exemple, si l'organisation évaluée est un club d'étudiants en génie logiciel, le vocabulaire utilisé dans la solution peut très bien être technique puisque tous les membres de l'organisation ont une certaine expérience en informatique. Un tel vocabulaire n'est pas à utiliser pour une organisation dont toutes les personnes ne sont pas pourvues d'une telle formation.

Cette présentation de l'organisation évaluée doit permettre d'obtenir des informations relatives à sa description générale (date de création, taille,...), à son personnel (nombre d'employés, connaissances des employés,...) à ses activités (nombre de projets en cours, durée moyenne d'un projet,...) ainsi que toutes autres informations s'avérant utiles pour mener au mieux la démarche d'amélioration. Une telle structure pourrait alors être représentée comme suit :

**Description de l'entreprise**

1. Pouvez-vous me présenter brièvement votre société ?
   - Nom :
   - Président :
   - Type :
   - Secteur d'activités :
   - Date de création :
   - Localisation :
   - Site web :
2. Pouvez-vous nous décrire brièvement le responsable de la démarche d'amélioration ?
   - Nom :
   - Prénom :
   - Formation :
   - Téléphone :
   - E-mail :
3. Pouvez-vous donner de plus amples informations sur votre personnel (Nombre d'employés, domaines de connaissances, formations,...)
4. Pouvez-vous nous décrire le contexte dans lequel évolue votre service à l'heure actuelle ? Quels sont les principaux projets en cours, quelles sont les priorités, la durée moyenne d'un projet, le nombre de projets en cours,...)

## 1.6 Représentation du questionnaire

La dernière recommandation pouvant être développée concerne la représentation du questionnaire. A l'heure actuelle, celui-ci est représenté sous format tableau via le programme Excel de Microsoft. Le problème avec un tel format est que la génération du rapport n'est pas automatique. Il faut en effet reprendre les différentes réponses aux questions et sous-questions et les insérer ensuite dans un rapport. Ce travail pourrait très bien se faire automatiquement une fois l'interview terminée.

Un autre désavantage à cette représentation est que le questionnaire n'est pas très portable puisqu'il faut avoir installé un programme permettant de gérer le tableur servant de structure au questionnaire. De plus, il est assez facile de le modifier, surtout sur la façon dont celui-ci calcule les scores attribués à chaque axe de la micro-évaluation. Ces scores étant nécessaires pour les tableaux synoptiques placés en fin de rapport.

Un dernier désavantage lié à la version actuelle de la micro-évaluation est son utilisation. Ce modèle étant utilisé dans le domaine informatique et ayant fait plus que ses preuves à l'heure actuelle, il pourrait être utile de le représenter davantage comme un outil que l'on utilise. Ainsi, l'utilisation d'une telle méthode se ferait plus facilement et serait moins lourde durant l'interview entre l'évaluateur et le représentant de l'entreprise et surtout, lors de la génération du rapport servant de base aux processus d'amélioration possibles.

Cependant, les avantages apportés par la version Excel de la micro-évaluation, comme par exemple la facilité de modifier les questions et sous-questions, si on veut passer du français à l'anglais, ne sont pas à mettre de côté.

## 1.7 Conclusion

Une fois les différentes améliorations décrites et mises sur pied, celles-ci ont été retranscrites au format papier. Cette retranscription papier est présentée dans la partie annexe de ce mémoire.

## Cinquième partie

# Outil statistique pour la Micro-Évaluation

# Section 1 :

## *Présentation de l'outil*

**Contenu :** *Présentation de l'outil statistique basé sur la micro-évaluation.*

### 1.1 Introduction

La micro-évaluation est maintenant utilisée dans divers pays, à savoir la Belgique, la France et le Canada. Ainsi, il devient plus qu'intéressant de pouvoir élaborer des statistiques sur base des résultats obtenus par cette méthode dans ces différents pays. Bien sûr, il est clair que ces statistiques peuvent être calculées via certains programmes tels qu'Excel. Cependant, la réalisation de ces calculs est lourde et longue puisqu'il faut au préalable enregistrer les différents scores sur lesquelles on va baser les statistiques.

C'est pourquoi, en collaboration avec Monsieur Naji Habra de l'Institut d'Informatique (FUNDP) et Monsieur Simon Alexandre du CETIC, un outil permettant d'élaborer des statistiques tout en allégeant la phase d'enregistrement des différents scores a été élaboré. C'est cet outil qui est présenté dans cette dernière partie de mémoire.

### 1.2 Exigences des clients

Avant de commencer tout document nécessaire à une bonne implémentation de l'outil, une prise de connaissance des diverses exigences spécifiées par les futurs utilisateurs de l'outil est nécessaire. Ces exigences furent les suivantes :

#### Exigences fonctionnelles

L'outil doit permettre de calculer des statistiques sans prendre en compte la version du questionnaire de la micro-évaluation, ce qui permet ainsi de ne pas trop restreindre l'échantillon sélectionné pour les calculs. Cette exigence est en partie remplie par un gestion statistique des axes de la micro-évaluation. Afin de calculer ces statistiques, l'outil doit permettre à l'utilisateur d'enregistrer les scores obtenus par une évaluation, quelque soit la version de son questionnaire.

### Exigences non fonctionnelles

- Premièrement, l'outil doit être le plus évolutif possible c'est-à-dire pouvoir travailler avec des versions différentes de la micro-évaluation et pouvoir évoluer facilement d'une langue à l'autre sans devoir réécrire tout l'outil. En ce qui concerne l'évolution du questionnaire, celle-ci a été limitée aux questions, le nombre des axes restant quant à lui fixe. On peut donc, d'une version à l'autre du questionnaire, avoir un nombre différent de questions ainsi qu'une répartition différente de celles-ci parmi les axes.
- Deuxièmement, il doit être accessible de n'importe quel ordinateur tant que celui-ci possède une connexion Internet.
- Troisièmement, il doit être sécurisé à un niveau minimum. Une authentification est donc nécessaire au démarrage de l'outil.
- Finalement, celui-ci doit être le plus documenté possible afin qu'il puisse continuer à évoluer sans perdre trop de temps à comprendre le code de celui-ci.

Sur base de ces différentes exigences, un outil statistique tel que présenté dans le point suivant a pu être élaboré.

## 1.3 Présentation générale

L'outil que nous avons élaboré est un outil statistique basé sur les scores obtenus en appliquant la méthode de la micro-évaluation sur diverses entreprises. Celui-ci est plus communément appelé MicroStat. Il comporte deux fonctionnalités principales : L'enregistrement de scores obtenus par application de la micro-évaluation sur une entreprise et le calcul des diverses statistiques sur les scores enregistrés[1].

Avant de passer à la description du moteur statistique proprement dit, les principales fonctionnalités de l'outil vont maintenant être présentées[2] :

### Enregistrement de scores

Cette première fonctionnalité représente la base du système puisque sans scores enregistrés au sein de celui-ci, le calcul de statistiques est rendu impossible. Comme le montre la figure ci-dessous, cet enregistrement consiste à spécifier le score obtenu pour chaque question du questionnaire avec lequel l'évaluation a été réalisée. Notons qu'avant cette étape, l'utilisateur devra enregistrer différentes informations relatives à l'entreprise évaluée, celles-ci incluant la version du questionnaire utilisé[3].

[1] Les calculs statistiques offerts par MicroStat sont définis dans le point suivant.

[2] Pour de plus amples informations, veuillez consulter le manuel d'aide placé en annexe.

[3] A ce stade, l'outil ne comporte que la dernière version de la micro-évaluation. Celle-ci est présente en annexe. Comme il est spécifié dans la partie consacrée à l'implémentation de MicroStat, d'autres versions peuvent être ajoutées.

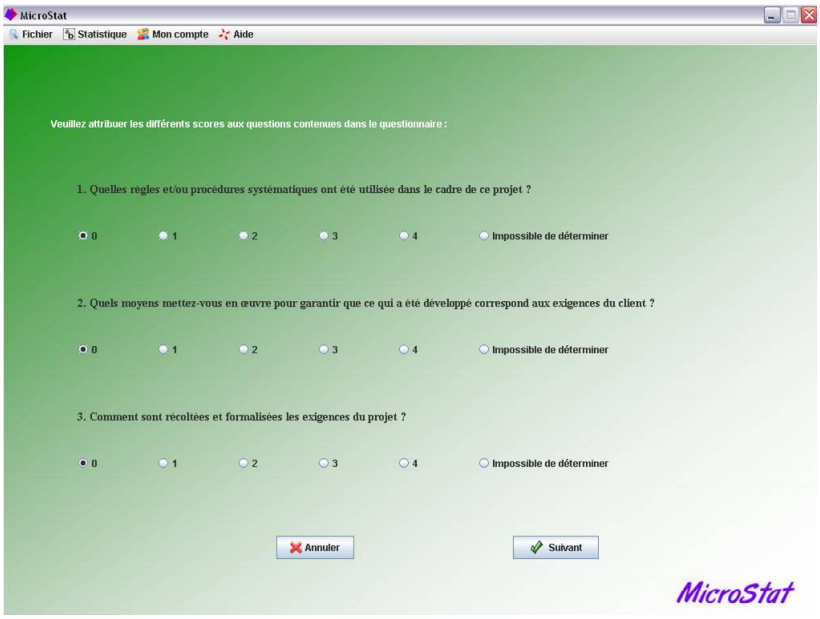



FIG. 1.1 – Enregistrement des scores dans MicroStat

## Calcul de statistiques

Cette seconde fonctionnalité représente quant à elle le coeur de l'outil puisqu'elle en est la principale. Comme le montre la figure ci-dessous, le calcul des statistiques peut être basé soit sur les axes, ce qui permet de calculer des statistiques plus générales et sans tenir compte des versions de questionnaire, ou soit sur les questions. Dans le cas des questions, l'utilisateur devra alors spécifier la version du questionnaire sur laquelle il veut baser ses statistiques.

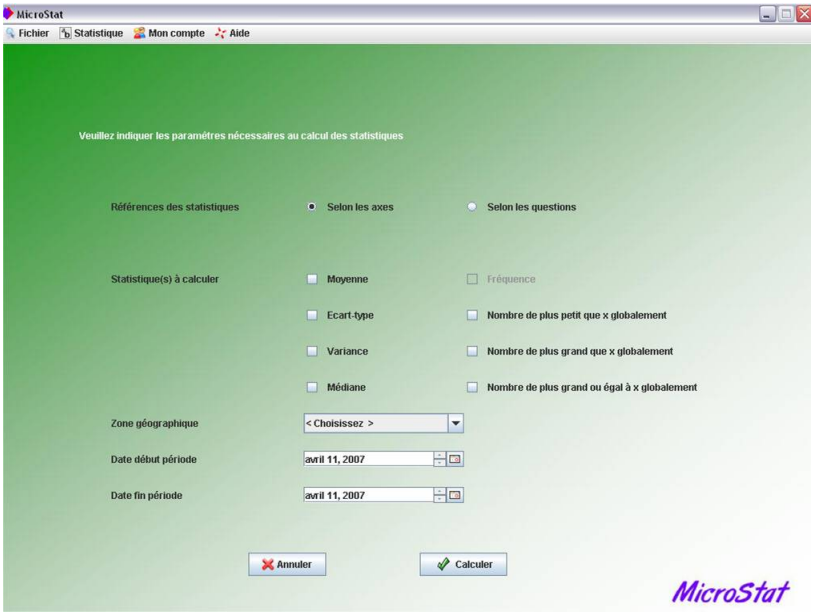



FIG. 1.2 – Calcul de statistiques dans MicroStat

Enuiste, il ne reste plus qu'à spécifier les calculs statistiques désirés, ainsi que la zone géographique, la date de début et de fin de la période sur laquelle les statistiques doivent être réalisées.

Une fois le calcul des statistiques effectué, l'échantillon et les calculs réalisés dessus sont enregistrés dans un fichier Excel permettant ainsi à l'utilisateur d'élaborer des statistiques n'étant pas encore prises en charge par le programme ou, de sortir des graphiques basés sur les statistiques calculées.

## 1.4 Moteur statistique

La fonctionnalité principale de l'outil MicroStat étant la gestion de divers calculs statistiques, un moteur statistique a été élaboré. Celui-ci permet de calculer les statistiques suivantes :

### Moyenne

*"On calcule la moyenne d'une variable numérique en additionnant les valeurs de toutes les observations incluses dans un ensemble de données, puis en divisant cette somme par le nombre d'observations qui font partie de l'ensemble. Ce calcul permet d'obtenir la valeur moyenne de toutes les données"* [30]. Prenons l'échantillon suivant : 0, 1, 2, 2, 3, 4. On obtiendra alors la moyenne suivante :

$$\begin{aligned} \textbf{Moyenne} &= (0 + 1 + 2 + 2 + 3 + 4) \ / \ 6 \\ &= 12/6 \\ &= 2 \end{aligned}$$

### Médiane

*"Si les observations d'une variable sont ordonnées par valeur, la valeur médiane correspond à l'observation qui se trouve au point milieu de cette liste ordonnée. Elle correspond plus précisément à un pourcentage cumulé de 50% (c'est-à-dire que 50% des valeurs sont supérieures à la médiane et 50% lui sont inférieures). La position de la médiane est donc la valeur (n+1)/2, le* ***n*** *désignant le nombre de valeurs dans un ensemble de données"* [30].

Pour calculer la médiane, il faut donc d'abord classer les données par ordre croissant. Prenons l'échantillon suivant : 0, 1, 2, 2, 3. On obtiendra alors la médiane suivante :

$$\begin{aligned} \textbf{Médiane} &= (5+1) \ / \ 2 \\ &= 6/2 \\ &= 3 \end{aligned}$$

La troisième valeur de l'échantillon de données sera la médiane, c'est-à-dire 2. Maintenant, prenons le cas d'un échantillon composé de 6 éléments : 0, 1, 2, 2, 2, 3. Si on effectue le calcul

de la médiane de la même manière que précédemment, on obtiendra 3,5. Ce résultat ne permet pas d'établir une médiane distincte. Dans ce cas, la médiane est calculée en faisant la moyenne du temps au-dessous de la médiane et du temps au-dessus de celle-ci :

$$\begin{aligned} \textbf{Médiane} &= \text{(Troisième valeur + quatrième valeur) / 2} \\ &= (2+2)/2 \\ &= 2 \end{aligned}$$

## Écart-type

*"L'écart-type est la mesure de dispersion la plus couramment utilisée en statistique lorsqu'on emploie la moyenne pour calculer une tendance centrale. Il mesure donc la dispersion autour de la moyenne En raison de ses liens étroits avec la moyenne, l'écart-type peut être grandement influencé si cette dernière donne une mauvaise mesure de tendance centrale"* [30].

*"L'écart-type est aussi influencé par les valeurs aberrantes ; une seule de ces valeurs pourrait avoir une grande influence sur les résultats de l'écart-type. Il s'agit donc d'un bon indicateur de l'existence de valeurs aberrantes, ce qui en fait une mesure de dispersion très utile pour les distributions symétriques ne comptant aucune valeur aberrante, tel que dans la micro-évaluation"*[30]. L'écart-type peut donc être calculé via la formule suivante :

$$\text{Écart-type} = \sqrt{\frac{\sum (x-\overline{x})^2}{n}}$$

Prenons comme exemple l'échantillon suivant : 1, 2, 2, 3, de moyenne 2. On obtiendra alors comme écart-type :

$$\begin{aligned} \textbf{Écart-type} &= \sqrt{((1-2)^2 + (2-2)^2 + (2-2)^2 + (3-2)^2)/4} \\ &= \sqrt{(1+0+0+1)/4} \\ &= \sqrt{2/4} \\ &= 0{,}7 \end{aligned}$$

## Variance

*"La variance est une mesure du degré de dispersion d'un ensemble de données. On la calcule sous la forme de l'écart au carré moyen de chaque nombre par rapport à la moyenne d'un ensemble de données"* [30]. Prenons comme exemple l'échantillon 1, 2, 2, 3, de moyenne 2 :

$$\begin{aligned} \textbf{Variance} &= [(1-2)^2 + (2-2)^2 + (2-2)^2 + (3-2)^2]/4 \\ &= (1+0+0+1)/4 \\ &= 0{,}5 \end{aligned}$$

Notons qu'une fois l'écart-type calculé, on peut facilement en découler la variance puisque l'écart-type équivaut au carré de la variance, et inversement.

### Fréquence

*"La fréquence d'une observation particulière est le nombre de fois que l'observation se dégage des données. La distribution d'une variable est le profil des valeurs de l'observation. Les distributions de fréquences peuvent être représentées sous forme de tableaux ou de diagrammes"* [30]. Prenons l'exemple 0,1,4,3,1,0,3,3,2,0,1 reflétant le nombre de voiture par ménage. On obtiendra alors le tableau de fréquences suivant :

| **Score** | **Comptage** | **Fréquence** |
|---|---|---|
| 0 | *** | 3 |
| 1 | *** | 3 |
| 2 | * | 1 |
| 3 | *** | 3 |
| 4 | * | 1 |

En examinant rapidement ce tableau de distribution de fréquences, il peut être constaté que sur les 11 ménages sondés, 3 ne possèdent pas de voiture, 3 en possèdent une,...

### Nombre de résultat suivant une condition x

Cette statistique détermine le nombre d'éléments contenus dans l'échantillon sélectionné vérifiant la condition x. Dans le cas de l'outil statistique MicroStat, la condition x peut être de trois types différents, ce qui donne trois calculs statistiques possibles :

1. Nombre de résultats plus petits que x, x $\in$ {1,2,3,4}
2. Nombre de résultats plus grands que x, x $\in$ {0,1,2,3}
3. Nombre de résultats plus grands ou égaux à x, x $\in$ {1,2,3,4}

Afin d'avoir une idée plus claire du fonctionnement de ces trois calculs, prenons l'échantillon suivant : 0,0,1,1,1,2,3,3,3,3,4. Les différents tableaux suivants sont alors obtenus :

*a) Nombre de résultats plus petits que x, x $\in$ {1,2,3,4}*

| **< 1** | **< 2** | **< 3** | **< 4** | **Total** |
|---|---|---|---|---|
| 2 | 5 | 6 | 10 | 11 |

*b) Nombre de résultats plus grands que x, x $\in$ {0,1,2,3}*

| **Total** | **> 0** | **> 1** | **> 2** | **> 3** |
|---|---|---|---|---|
| 11 | 9 | 6 | 5 | 1 |

*c) Nombre de résultats plus grands ou égaux à x, $x \in \{1,2,3,4\}$*

| **Total** | $\geq$ **1** | $\geq$ **2** | $\geq$ **3** | $\geq$ **4** |
|---|---|---|---|---|
| 11 | 9 | 7 | 5 | 1 |

Via ces différents tableaux, il sera ainsi possible d'établir différents graphiques mettant en avant diverses comparaisons au point de vue des axes de la micro-évaluation ou encore au point de vue des questions de celle-ci.

## 1.5 Conclusion

L'outil statistique MicroStat ayant été maintenant brièvement présenté, l'analyse des besoins telle que présentée dans la section suivante peut alors être présentée. Après celle-ci, l'implémentation de l'outil et les travaux futurs qui peuvent être réalisés sur celui-ci seront présentés.

# Section 2 :

## *Analyse des besoins*

> **Contenu :** *Présentation de la phase d'analyse des besoins de l'outil MicroStat c'est-à-dire la phase d'analyse nécessaire à sa bonne implémentation.*

Les différents points de l'analyse des besoins qui vont être développés sont la description des classes d'utilisateurs, les fonctionnalités du système ainsi que la représentation de la statique des données.

### 2.1 Classes d'utilisateurs

L'outil statistique comporte deux classes d'utilisateurs : l'administrateur et l'utilisateur. Il est à noter que pour avoir accès aux fonctionnalités non restreintes du système, l'administrateur doit s'enregistrer comme utilisateur. Celles-ci se décrivent de la façon suivante :

#### L'administrateur

Le but du travail de l'administrateur est de mettre à jour le système en y incluant et en y supprimant les utilisateurs qui doivent faire l'objet d'une telle tâche. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel. Du fait de sa qualification et de son intérêt évident pour la micro-évaluation et le calcul de statistiques relatives à celle-ci, l'administrateur adopte une attitude positive vis-à-vis de la tâche. Sa langue maternelle est le français.

Les périphériques d'entrée utilisés sont le clavier et la souris. Le système d'exploitation de l'utilisateur est Windows. Il n'y a pas de versions types de celui-ci (Professionnel ou Familial). Une machine virtuelle Java doit être installée. L'administrateur utilise le logiciel seul et sur tout ordinateur possédant une connexion Internet.

#### L'utilisateur

Le but du travail de l'utilisateur est de mettre à jour le système en y incluant les nouveaux résultats obtenus par les évaluations qu'il produit. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel. Du fait de sa qualification et de son intérêt évident pour la

micro-évaluation et le calcul de statistiques relatives à celle-ci, l'utilisateur adopte une attitude positive vis-à-vis de la tâche. Sa langue maternelle est le français.

Les périphériques d'entrée utilisés sont le clavier et la souris. Le système d'exploitation de l'utilisateur est Windows. Il n'y a pas de versions types de celui-ci (Professionnel ou Familial). Une machine virtuelle Java doit être installée. Le logiciel est utilisé seul et sur tout ordinateur possédant une connexion Internet.

## 2.2 Les fonctionnalités du système

Les Use Cases représentent le comportement affiché par le système sous certaines conditions, de manière à satisfaire un objectif de l'un des acteurs. Les Use Case Diagrams montrent quant à eux les acteurs, les limites du système, les relations entre les acteurs et le système ainsi que les relations entre les Use Cases eux-mêmes. Ceux-ci permettent, aux travers de scénarii, de capturer, de représenter et de valider les fonctions d'un domaine, de les spécifier et de donner un aperçu de la dynamique et des interactions. Du fait de leur simplicité, les Use Cases ont pour avantage notable d'être compréhensibles par tous.

Les fonctionnalités principales du système peuvent ainsi être représentées par les deux diagrammes suivant :

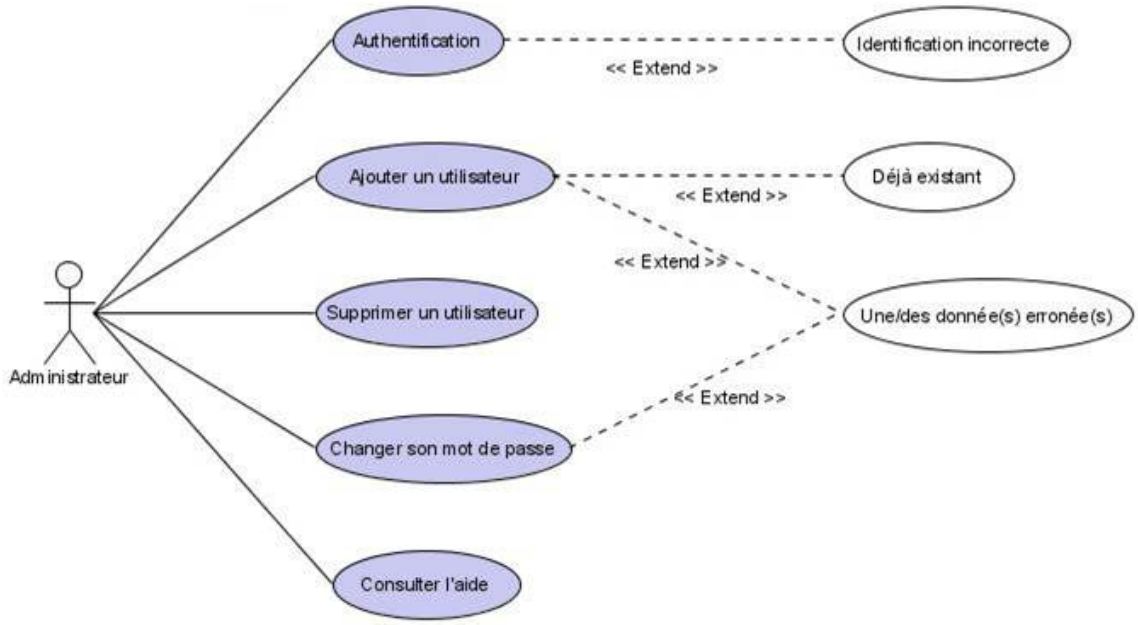



FIG. 2.1 – Diagramme Use Case de l'administrateur

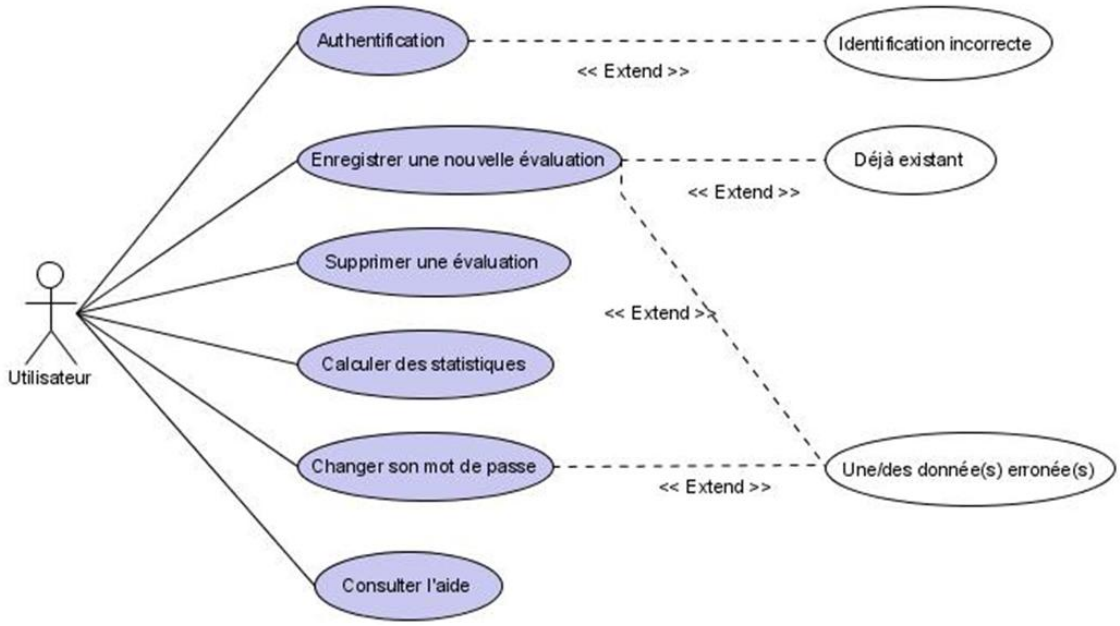



Fig. 2.2 – Diagramme Use Case de l'utilisateur

### Authentification

L'authentification apporte une sécurité minime à l'outil statistique de la micro-évaluation. En effet, lors du lancement du programme, l'utilisateur ou l'administrateur se voit obligé d'entrer un login et un mot de passe valide afin de pouvoir rentrer au sein de celui-ci.

### Enregistrer une nouvelle évaluation

L'enregistrement des résultats d'une nouvelle évaluation réalisée dans une entreprise représente la première fonctionnalité importante du futur outil. Cette fonctionnalité consiste à enregistrer, pour chaque question reprise dans le questionnaire de la micro-évaluation, les différents scores obtenus.

### Supprimer une évaluation

La suppression d'une évaluation déjà enregistrée dans le système peut être réalisée par un utilisateur dans le cas où cette évaluation n'est plus nécessaire ou si elle est fausse. Dans ce cas, toute information relative à cette évaluation sera supprimée.

### Calculer des statistiques

La deuxième fonctionnalité importante du futur système consiste à permettre à l'utilisateur de calculer un ensemble de statistiques sur base des différents scores enregistrés dans le

système (Variance, Écart-type, Moyenne,...). Cependant, vu que le questionnaire de la micro-évaluation est en constante évolution, le calcul de statistiques ne peut se réaliser qu'avec des scores prévenants d'une version identique de questionnaire.

### Ajouter un utilisateur

Seul l'administrateur peut accéder à cette fonctionnalité qui lui permettra d'enregistrer un nouvel utilisateur dans le système. Lors de cet enregistrement, il devra spécifier certaines informations sur le nouvel utilisateur tels que son nom, son prénom, son mot de passe, ...

### Supprimer un utilisateur

La suppression d'un utilisateur enregistré dans le système ne peut être entreprise que par un administrateur. L'utilisateur ainsi supprimé verra son accès au système refusé lors de sa prochaine connexion.

### Changer son mot de passe

Cette fonctionnalité permet à tout utilisateur qui le souhaite ou à l'administrateur de modifier son mot de passe actuel. Une fois réalisée, seul le nouveau mot de passe permettra à l'utilisateur ou à l'administrateur de rentrer à nouveau dans le système.

### Consulter l'aide

La consultation de l'aide représente la fonctionnalité la plus basique du système. Son objectif est de fournir, à l'utilisateur ou à l'administrateur, le support nécessaire lors de toutes incompréhensions dans l'utilisation de l'outil statistique.

## 2.3 Représentation de la statique des données

L'outil s'appuie sur une base de données MySql installée sur leur serveur. Celle-ci peut être représentée via un modèle Entité-Relation-Attribut. Ce modèle propose des concepts, principalement les entités, les associations et les attributs, permettant de décrire un ensemble de données relatives à un domaine défini. On obtient alors le schéma suivant :

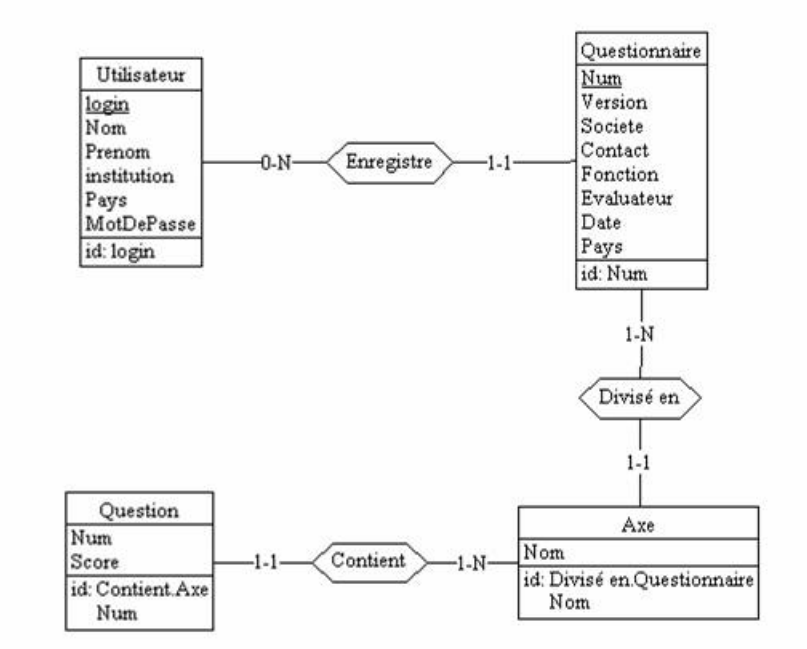



FIG. 2.3 – Représentation de la statique des données

Le schéma ERA ci-dessus contient 4 entités qui peuvent être décrites de la façon suivante :

### Utilisateur

Cette entité contient les utilisateurs enregistrés dans le système. Chaque utilisateur est identifié par un login numérique à valeur unique qui est généré automatiquement lors de son enregistrement dans le système.

### Questionnaire

L'entité Questionnaire contient l'ensemble des informations contenues dans les questionnaires résultant des évaluations et qui sont nécessaires au système. Chaque questionnaire est identifié par un numéro de questionnaire unique généré automatiquement.

### Axe

Un questionnaire étant divisé en minimum un axe, cette entité contient les différents axes contenus dans les questionnaires. Chaque axe est identifié par un numéro et par le numéro du questionnaire auquel il fait référence.

### Question

Chaque axe étant divisé en minimum une question. Cette entité contient donc les différentes questions contenues dans les différents axes. Chaque question est identifiée par un numéro, par le nom de l'axe auquel elle se rapporte et par le numéro de questionnaire auquel l'axe se rapporte.

## 2.4 Approbation du cahier des charges

Le cahier des charges ne représente que les fonctionnalités premières de l'outil. Il est clair de proposer d'aller plus loin dans son élaboration si le temps le permet. Il est ainsi possible d'ajouter des fonctionnalités secondaires en fonction du temps restant.

Ce cahier des charges a été approuvé par les différents représentants des utilisateurs futurs de l'outil, à savoir Monsieur Naji Habra de l'Institut d'Informatique (FUNDP) et Monsieur Simon Alexandre du CETIC. Ainsi, l'implémentation de l'outil a pu être entreprise et est présentée dans la section suivante.

# Section 3 :

# *Implémentation de l'outil*

**Contenu :** *Description de la phase d'implémentation de l'outil MicroStat.*

## 3.1 Choix d'implémentation

Sur base du cahier des charges décrit dans la section précédente et des exigences spécifiées par le client telles que reprises dans la présentation de l'outil, il a été proposé de réaliser l'outil au moyen d'une base de données MySql et du langage Java. Le choix de ce langage reste ainsi dans la continuité logique du stage puisqu'il a été utilisé plusieurs fois dans le cadre de l'élaboration de divers outils, afin d'améliorer les processus et pratiques logiciels de plusieurs entreprises. Grâce aux expériences et aux connaissances issues de ce stage, il sera possible d'établir un outil comportant une base solide, facilement évolutif et mettant en valeur tout notre savoir-faire.

Ce langage de programmation permet de répondre aux différentes exigences du client, comme par exemple, fournir un code clair, structuré et documenté au moyen de la Javadoc. Le seul inconvénient est qu'il ne permet pas l'accès via une page web. Cependant, certaines techniques permettent de pallier ce problème sans trop de contraintes telle qu'une connexion SSH (Secure Shell) avant d'utiliser l'outil.

Au point de vue de la base de données, le langage utilisé est le MySQL et ce, pour plusieurs raisons. MySQL un serveur de bases de données relationnelles SQL très rapide, multi-thread, stable et multi-utilisateurs. MySQL fonctionne sur beaucoup de plates-formes différentes comme Windows, Linux et Mac OS X. Ses principaux avantages sont sa rapidité, sa robustesse et sa facilité d'utilisation [19]. C'est pourquoi, ce type de base de données a été choisi. Cependant, MySQL est un peu court en fonctionnalités, mais ce défaut ne pose pas problème pour MicroStat.

Devant réaliser un outil évolutif, différents fichiers XML (Extensible Markup Language) ont été utilisés. Ainsi, il est assez facile d'ajouter une nouvelle langue dans le programme en éditant un nouveau fichier XML tel que celui associé à la langue par défaut du programme (Français). De même, l'ajout d'une nouvelle version de questionnaire, contenant une répartition ou un nombre de questions différent, peut être facilement réalisé en éditant un nouveau fichier XML tel que celui dédié à la version par défaut du questionnaire.

## 3.2 Structure de l'outil

La structure de l'outil statistique MicroStat va maintenant être décrite via un schéma d'architecture de haut niveau et un diagramme de composants.

### 3.2.1 Architecture de l'outil

MicroStat est basé sur l'architecture de type "repository". Celle-ci se compose de deux types de modules. Le premier type est un repository centralisé où se trouvent les données partagées avec un modèle stable et un accès standard (Partie grisée de la figure ci-dessous). Le deuxième est un ensemble de composants actifs quasi-autonomes bâtis autour du premier (Parties non grisées de la figure ci-dessous). C'est le système informatique classique avec une base de données centralisée. On peut alors représenter l'outil par le schéma suivant, les cercles représentants les gestionnaires principaux de MicroStat :

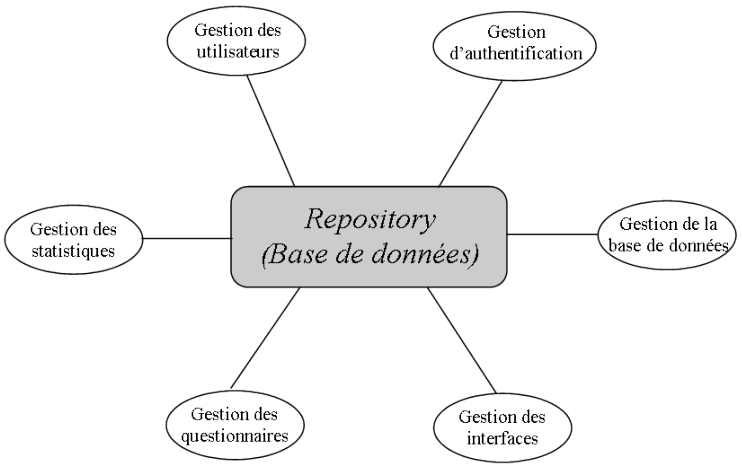



FIG. 3.1 – Architecture de l'outil statistique MicroStat

Ce style d'architecture a été choisi pour différentes raisons telles que sa simplicité, son efficacité, son accès centralisé aux données permettant un haut niveau de sécurité, une bonne sauvegarde des données, ... De plus, le repository ayant été installé sur un serveur, une application multi-utilisateurs a ainsi été développée où le serveur est joué par le repositiry tout en étant un module passif n'offrant qu'un accès centralisé aux données et où les clients sont joués par les différentes exécutions du gestionnaire de revue de code (chaque exécution se faisant sur un poste différent).

### 3.2.2 Diagramme de composants

Le diagramme de composants décrit l'organisation du système du point de vue des modules de code. Ce diagramme permet de mettre en évidence les dépendances entre les composants (qui utilise quoi) et permet ainsi de mieux organiser les modules. L'encapsulation (insérer un composant dans un autre, cacher des détails d'implémentation,...) permet de réduire la complexité et d'élever le niveau d'abstraction. MicroStat peut donc être représenté par le diagramme de composants suivant :

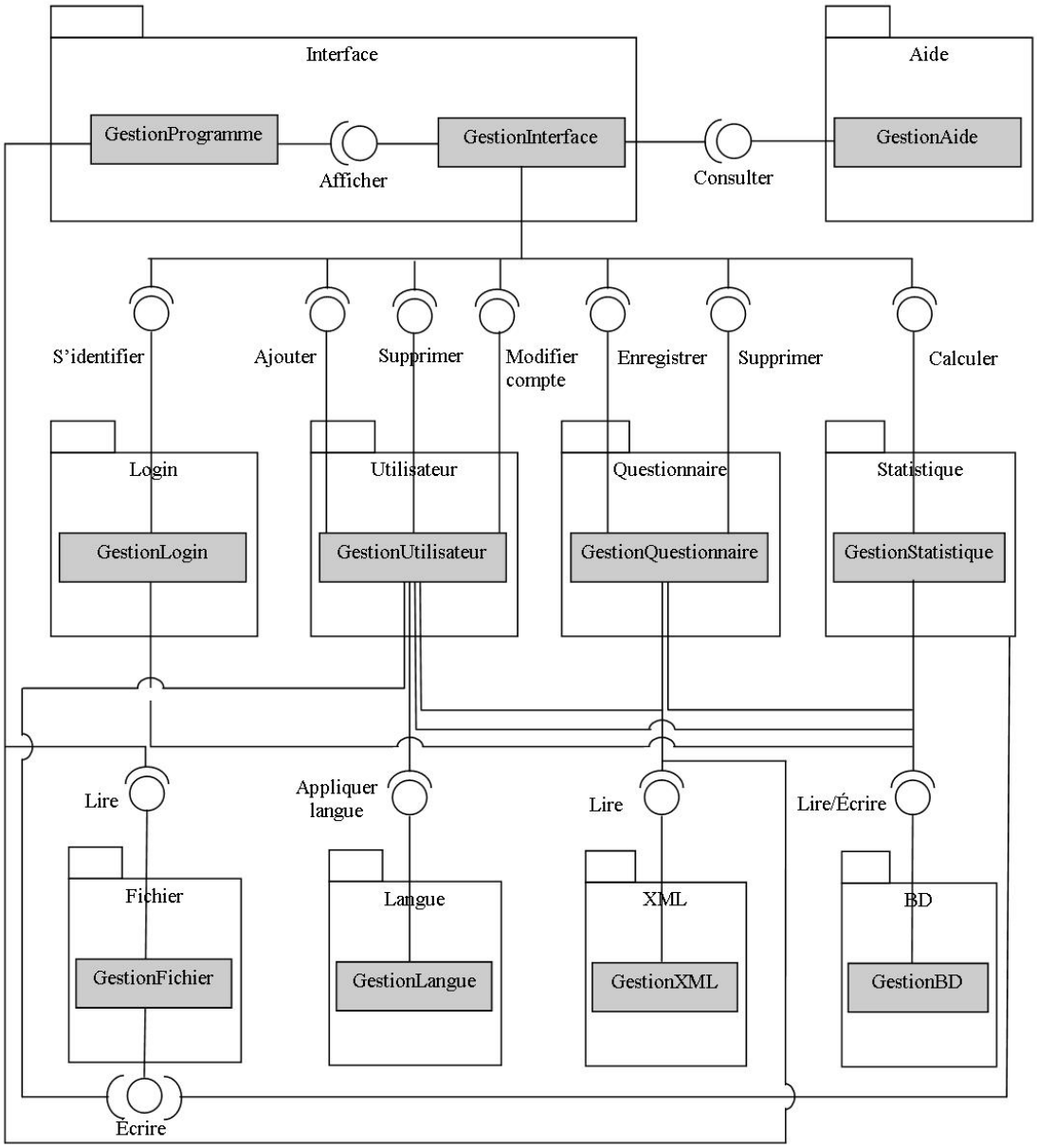



FIG. 3.2 – Diagramme de composants de MicroStat

Afin de bien comprendre la manière dont le programme a été implémenté, les différentes fonctionnalités de l'outil sont décrites pas à pas sur base du diagramme de composants tel que présenté ci-dessus. Cependant, il est d'abord nécessaire d'expliquer ce que le programme réalise lors de son chargement. En effet, avant d'afficher l'interface, le gestionnaire "GestionProgramme" va d'abord charger le programme par la langue par défaut de l'utilisateur. Afin de connaître cette langue par défaut, le programme va lire le fichier de propriétés de l'utilisateur, ce qui lui permettra aussi de se connecter à la base de données. Une fois ces différentes actions réalisées, le gestionnaire passera la main au gestionnaire "GestionInterface" offrant à l'utilisateur la possibilité de s'identifier.

Pour permettre à l'utilisateur d'entrer dans MicroStat, le gestionnaire GestionLogin va collaborer avec le gestionnaire GestionBD afin de vérifier que le login et le mot de passe de l'utilisateur sont corrects. Le gestionnaire redonne ensuite la main à l'appelant. Une fois dans le programme, l'utilisateur se verra offrir les fonctionnalités suivantes :

### Ajouter ou Supprimer un utilisateur

L'ajout et la suppression d'un utilisateur utilisent le même gestionnaire pour réaliser leur tâche. En effet, dans les deux cas, le gestionnaire "GestionUtilisateur" collaborera avec le gestionnaire "GestionBD" afin d'ajouter ou de supprimer l'utilisateur spécifié. Le gestionnaire appelé repassera ensuite la main au gestionnaire "GestionInterface" afin de permettre à l'utilisateur d'appeler une autre fonctionnalité.

### Modifier son compte

Cette fonctionnalité permet à l'utilisateur de modifier son mot de passe et/ou sa langue par défaut. Dans le premier cas, le gestionnaire "GestionUtilisateur" collaborera avec le gestionnaire "GestionBD", tout comme dans les situations d'ajout et de suppression d'un utilisateur.

Dans le second cas, différents gestionnaires seront appelés : "GestionFichier" afin d'écrire la nouvelle langue par défaut dans le fichier de propriétés de l'utilisateur, "GestionXML" afin de lire le fichier XML correspondant à la nouvelle langue par défaut de l'utilisateur et, "GestionLangue" afin de charger le programme de la nouvelle langue par défaut de l'utilisateur via le résultat de lecture du fichier XML.

Dans les deux cas, le gestionnaire "GestionUtilisateur" repassera ensuite la main au gestionnaire "GestionInterface" afin de permettre à l'utilisateur d'appeler une autre fonctionnalité.

### Enregistrer des scores

Pour enregistrer des scores relatifs à l'évaluation d'une entreprise[1], le gestionnaire "GestionQuestionnaire" devra d'abord appeler le gestionnaire "GestionXML" afin de lire le fichier

[1]Cette fonctionnalité comporte deux étapes : premièrement, enregistrer les informations relatives à l'entreprise évaluée et deuxièmement, enregistrer les scores obtenus sur cette entreprise.

XML correspondant à la version du questionnaire de la micro-évaluation. Celle-ci est spécifiée par l'utilisateur lors de l'enregistrement des informations relatives à l'entreprise évaluée.

Ensuite, ce même gestionnaire appellera le gestionnaire "GestionBD" afin d'enregistrer les informations et scores de l'entreprise évaluée. Le gestionnaire "GestionQuestionnaire" repassera ensuite la main au gestionnaire "GestionInterface" afin de permettre à l'utilisateur d'appeler une autre fonctionnalité.

### Supprimer des scores

Pour supprimer des scores relatifs à une entreprise et enregistrés dans le système, le gestionnaire "GestionQuestionnaire" collaborera avec le gestionnaire "GestionBD" afin qu'il supprime les scores et informations concernés. Le gestionnaire "GestionQuestionnaire" repassera ensuite la main au gestionnaire "GestionInterface" afin de permettre à l'utilisateur d'appeler une autre fonctionnalité.

### Calculer des statistiques

Pour calculer des statistiques, le gestionnaire "GestionStatistique" collaborera avec le gestionnaire "GestionBD" afin de récupérer informations nécessaires au calcul des statistiques spécifiées par l'utilisateur. Une fois les statistiques calculées, il appellera le gestionnaire "GestionFichier" afin de créer le fichier Excel qui contiendra l'ensemble des données et statistiques calculées. Le gestionnaire "GestionStatistique" repassera ensuite la main au gestionnaire "GestionInterface" afin de permettre à l'utilisateur d'appeler une autre fonctionnalité.

### Consulter l'aide

La consultation de l'aide se réalise tout simplement via le gestionnaire "GestionAide" qui ouvrira le fichier d'aide à l'utilisateur. Le gestionnaire "GestionAide" repassera ensuite la main au gestionnaire "GestionInterface" afin de permettre à l'utilisateur d'appeler une autre fonctionnalité.

## 3.3 Conclusion

Ce diagramme de composants clôture la section relative à l'implémentation de MicroStat. Afin de terminer la partie recherche consacrée à cet outil statistique, les différents travaux futurs pouvant être réalisés sur MicroStat vont être passés en revue. Ceux-ci sont présentés dans la section suivante.

# Section 4 :

## *Travaux futurs*

**Contenu :** *Présentation des travaux futurs réalisables sur MicroStat.*

## 4.1 Introduction

Sur base des exigences des futurs utilisateurs de MicroStat, un outil statistique de base assez évolutif a été développé. Celui-ci permet actuellement d'enregistrer des scores provenant d'évaluations d'entreprises via la micro-évaluation et permet de calculer des statistiques. Cependant, afin d'obtenir un outil complet et vu que le temps ne le permettait pas, certains travaux supplémentaires peuvent être réalisés sur l'outil. Ceux-ci sont présentés dans cette dernière section.

## 4.2 Travaux futurs

Les travaux futurs recommandés dans le cadre d'amélioration de MicroStat sont au nombre de deux, à savoir :

### Gestion dynamique du nombre d'axes

En effet, l'outil gère pour le moment un nombre dynamique de questions. Ainsi, il est possible d'ajouter ou supprimer des questions, le tout étant géré par l'outil. Cependant, le nombre d'axes doit, quant à lui, rester statique, à savoir 6. Une gestion dynamique des axes telles que celle utilisée pour la gestion des questions nous semble être une bonne amélioration à apporter à l'outil.

Néanmoins, cette gestion dynamique des axes n'a pas été inclue dans l'outil pour diverses raisons. Premièrement, le questionnaire de la micro-évaluation a toujours évolué en gardant le même nombre et le même type d'axes et deuxièmement, le temps accordé à l'implémentation de l'outil ne permettait pas de prendre en compte cette gestion dynamique.

### Gestion de documentation

Quand un utilisateur enregistre des nouveaux scores obtenus pas une entreprise suite à l'utilisation de la micro-évaluation, il enregistre en même temps certaines caractéristiques relatives à celle-ci. Ces caractéristiques ( le nom de l'entreprise, le pays d'origine de l'entreprise, le contact au sein de l'entreprise et sa fonction, la date d'interview, la version de la micro-évaluation, ...) ne sont pour le moment pas toutes utilisées.

Il serait recommandé d'intégrer à l'outil un gestionnaire permettant de faire ressortir, sous format texte ou autre, ces différentes informations. De même, l'ajout de nouvelles informations à spécifier lors de l'enregistrement de score pourrait être envisageable afin de fournir une documentation complète.

## 4.3 Conclusion

Cette dernière section relative aux travaux futurs pouvant être réalisés sur MicroStat clôture ainsi la partie recherche du mémoire centrée sur cet outil. Nous espérons avoir fourni aux utilisateurs de l'outil une bonne base, facilement exploitable et qui continuera à évoluer en dehors de ce mémoire.

# Conclusion

A travers ce mémoire centré sur l'amélioration de la qualité des pratiques et processus logiciels via la micro-évaluation, nous avons d'abord passé en revue les différents modèles, normes et référentiels existant dans ce domaine. Après analyse de ceux-ci, nous avons constaté qu'ils n'étaient pas adaptables aux entreprises avec lesquelles nous allions travailler.

C'est pourquoi, nous avons opté pour le modèle OWPL tel que décrit dans la première partie de ce mémoire. En effet, celui-ci est adapté au contexte des petites et très petites entreprises. De plus, il comporte une méthode simple et rapide permettant d'évaluer la qualité des processus et pratiques logiciels d'une petite ou très petite entreprise : la micro-évaluation.

Sur base de cette méthode, nous avons élaboré une démarche permettant d'entreprendre l'amélioration d'une pratique dans une entreprise. Ainsi, nous avons pu entreprendre, dans des entreprises québécoises, différentes démarches d'amélioration : l'une basée sur l'amélioration de la vérification et de la validation des projets pour un club d'étudiants en robotique de l'École de Technologie Supérieure de Montréal et deux autres basées sur la formalisation des exigences et la gestion des changements d'exigences pour une entreprise œuvrant dans la milieu agro-environnemental.

Grâce à cette expérience, nous avons été capables de critiquer la version de la micro-évaluation que nous avions utilisée au Québec. En effet, celle-ci nous semblait trop simple et dans certains cas, pas assez précise. C'est pourquoi, nous avons ajouté à cette version un lexique, différentes sous-questions ainsi qu'une section du rapport concernant la présentation de l'entreprise évaluée. Ces différentes améliorations permettent ainsi, à la personne en charge de la démarche d'amélioration, de mieux comprendre l'entreprise ciblée et ses besoins.

Il est donc clair que le modèle OWPL ainsi que la micro-évaluation sont des outils ayant fait leurs preuves dans l'évaluation et l'amélioration des processus et pratiques logiciels d'une petite ou très petite entreprise. Ayant atteint un certain niveau de maturité, il est maintenant possible d'aller plus loin dans l'élaboration de ce modèle, en mettant, par exemple, sur pied différents outils permettant d'exploiter les résultats obtenus.

C'est pourquoi, en collaboration avec l'Institut d'Informatique et le CETIC, nous avons élaboré un outil statistique permettant d'analyser les scores obtenus par la micro-évaluation. Cet outil, présenté dans la dernière partie de ce mémoire, a été développé dans un cadre d'évolutivité totale, en d'autres mots, il gère différentes versions de questionnaire ainsi que différentes langues. Cependant, comme nous l'avons spécifié, nous n'avons qu'élaboré que les bases de l'outil statistique et il reste donc du travail à réaliser sur celui-ci afin qu'il soit des plus complets.

Ainsi, nous pensons avoir fourni aux utilisateurs de la micro-évaluation un outil leur permettant d'analyser leurs résultats, d'élaborer des comparaisons entre les années et les pays dans lesquels la micro-évaluation est utilisée.

# Bibliographie

### Articles consultés


[1] Ba Tuan-Anh Pham, Dubé Damien et Viau Mathieu, *Rapport de micro-évaluation des pratiques logicielles - Capra*, École de Technologie Supérieure, Montréal, Québec, Canada, 20 juillet 2006.

[2] Abou El Fattah Mohammad, *Rapport de micro-évaluation des pratiques logicielles - Horizon Vert*, École de Technologie Supérieure, Montréal, Québec, Canada, 25 juin 2004.

[3] René Bujold, *Ingénierie des exigences, l'outil GenSpec*, Hydro-Québec, Montréal, Québec, Canada, Automne 2005.

[4] René Bujold, *Ingénierie des exigences, Une méthode simple et systématique*, Hydro-Québec, Montréal, Québec, Canada, Automne 2004.

[5] Naji Habra (FUNDP), Alain Renault (CETIC), *OWPL - Une méthodologie et des Modèles Légers pour Initier une Démarche d'Amélioration des Pratiques logicielles APL*, Belgique.

[6] Anabel Stambollian (ETS) , Naji Habra (FUNDP), Claude Laporte et Jean-Marc Desharnais (ETS), Alain Renault (Centre de recherche Henri Tudor), *OWPL : A light Model & Methodology for initiating software process improvement*, Proceedings of SPICE Conference, Luxembourg, 4-5 Mai 2006.

[7] Anabel Stambollian, *L'amélioration de la performance des processus et pratiques logiciels dans les petites entreprises françaises*,ETS, Montréal, Québec, Canada, Avril 2006.

[8] Alain Renault, *Initiating SPI in small enterprises, experiments with the Micro-Evaluation framework*, CETIC, Mai 2005.

[9] Naji Habra, **Génie logiciel**, *Ingenierie des logiciels - Cours de 2e Maîtrise*, Version 2005-2006, FUNDP.

[10] Alain Renault, *OWPL Micro-Évaluation*, Décembre 2006, CETIC ASBL, ©2003 - FUNDP & CETIC, OWPL-Micro-Version.xls

[11] Claude Laporte, Alain Renault, Naji Habra, *Initiating Software Process Improvement in Small Enterprises : Experiments with Micro-Evaluation Framework*, mars 2006, Montréal, Canada.

[12] Software Engineering Institute, *Capability Maturity Model Integration (CMMI)* Carnegie Melon University, 2002.

[13] Borland, *SCAMPI : Méthode standard d'évaluation CMMI®pour l'amélioration des processus*, TeraQuest Inc, Janvier 2004.

[14] T.P. Rout, *The Rapid Assessment of Software Process Capability*, Software Quality Institute, Griffith University, Queensland, Australia, Mai 2000.

[15] Claude Laporte, Alain April, Alain Renault, *L'application de normes de génie logiciel dans les très petites entreprises : Historique et premiers résultats*, 2005, Génie Logiciel, No. 75, 7-12.

[16] A. Anacleto, C. G. von Wangenheim, C. F. Salviano et R. Savi, *Experiences gained from applying ISO/IEC 15504 to small software companies in Brazil*, 4th International SPICE

Conference on Process Assessment and Improvement, Lisbonne, Portugal, pp. 33-37, Avril 2004.

[17] N. Habra, A. Renault, *Evaluation et amélioration des pratiques logicielles dans les PME wallonnes - Positionnement et compatibilité par rapport aux normes ISO 9000*, Institut d'Informatique, FUNDP - Charleroi, Décembre 2000.

[18] N. Habra, A. Renault, *Evaluation et amélioration des pratiques logicielles dans les PME wallonnes - Version 1.2.2 b*, Institut d'Informatique, FUNDP - Charleroi, Janvier 2001.

[19] Raja Dallapé, *Re-ingénierie d'un système d'aide à la décision : Du raisonnement hybride au raisonnement par règle*, Institut d'Informatique, FUNDP, 2005-2006.

### Pages web consultées

[20] *Qu'est-ce que le SAAP ?*, `http://horizonvert.dyndns.org/accueil/saap.asp`

[21] *Wikipédia*, `http://fr.wikipedia.org`

[22] *SPICE*, `http://www.sqi.gu.edu.au`

[23] *What is SPICE ?*, `http://www.sqi.gu.edu.au/spice`

[24] *Introduction au méta modèle ISO 15504*, `http://www.rad.fr/spice0.htm`

[25] *Etude du "StandishGroup International, Inc." 1995, 365 organisations, 8380 projets*, `http://www.standishgroup.com/chaos.html`

[26] *SEI Capability Maturity Model*, `http://www2.umassd.edu/swpi/processframework/cmm/cmm.html`

[27] *Glossaire du guide de conduite de projet systèmes d'information*, `http://www.dsi.cnrs.fr/conduite-projet/glossaire.htm`

[28] *Roue de Deming*, `http://fr.wikipedia.org/wiki/Roue_de_Deming`

[29] *PDCA, Roue de Deming*, `http://www.quesaco.org/PDCA-Roue-de-Deming`

[30] *Calcul de statistiques*, `http://www.statcan.ca/francais.htm`

# Sixième partie

# Annexes

# Annexe A :

*CodeReview - Implémentation*

# Code Review - Implémentation

> **Contenu :** *Description de l'implémentation, de la vérification et de validation de l'outil destiné à Club Capra : Code Review.*

## Choix d'implémentation

Plusieurs choix d'implémentation ont dû être faits avant de commencer la conception du programme de gestion de revue de code. Le premier choix s'est porté sur le langage de programmation à utiliser. Celui-ci fut le java puisque les membres de Club Capra travaillent principalement avec ce langage et ont émis le souhait de pouvoir continuer à faire évoluer le programme via l'insertion de différents plug-in. Un autre choix fut fait sur l'exportation des documents. La technologie utilisée est celle des fichier pdf (**P**ortable **D**ocument **F**ormat) pour leur portabilité et leur facilité de lecture (Aucun problème de version ou de licence pour les lire). Le programme a été implémenté sous Eclipse.

Pour gérer la base de données, nous avons travaillé avec le langage MySql et ce pour plusieurs raisons : sa robustesse, sa gestion multi-utilisateurs et surtout parce que les logiciels pour gérer une base de données MySql sont libres et gratuits. La base de données a été créée avec MySql Server et gérée avec Aqua Data Studio.

Sera décrite ci-dessous la méthode d'implémentation pour élaborer le gestionnaire de revue de code en commençant par la représentation du système et ensuite une explication brève de la phase de test.

## Validation des exigences

Une fois les choix d'implémentation effectués, et dans le but de valider les exigences du client et donc de construire un programme complet et adapté, nous avons réalisé un prototype d'interface représentant graphiquement les différentes fonctionnalités du programme. Cette méthode de validation des exigences permet d'obtenir directement l'avis du client sur le programme. En effet, il peut alors vous faire part de ses commentaires sur la manière dont vous représentez ses exigences : ajout d'informations supplémentaires, modification de la manière de procéder pour représenter une fonctionnalité du programme, ...

Dans le cas de Club Capra, cette étape de validation des exigences a été très intéressante. Les dirigeants du club ont pu exprimer leur avis sur la manière dont le programme pouvait être réalisé. Ainsi, certaines idées et éléments ont pu être rajoutés au système. Par exemple, gérer

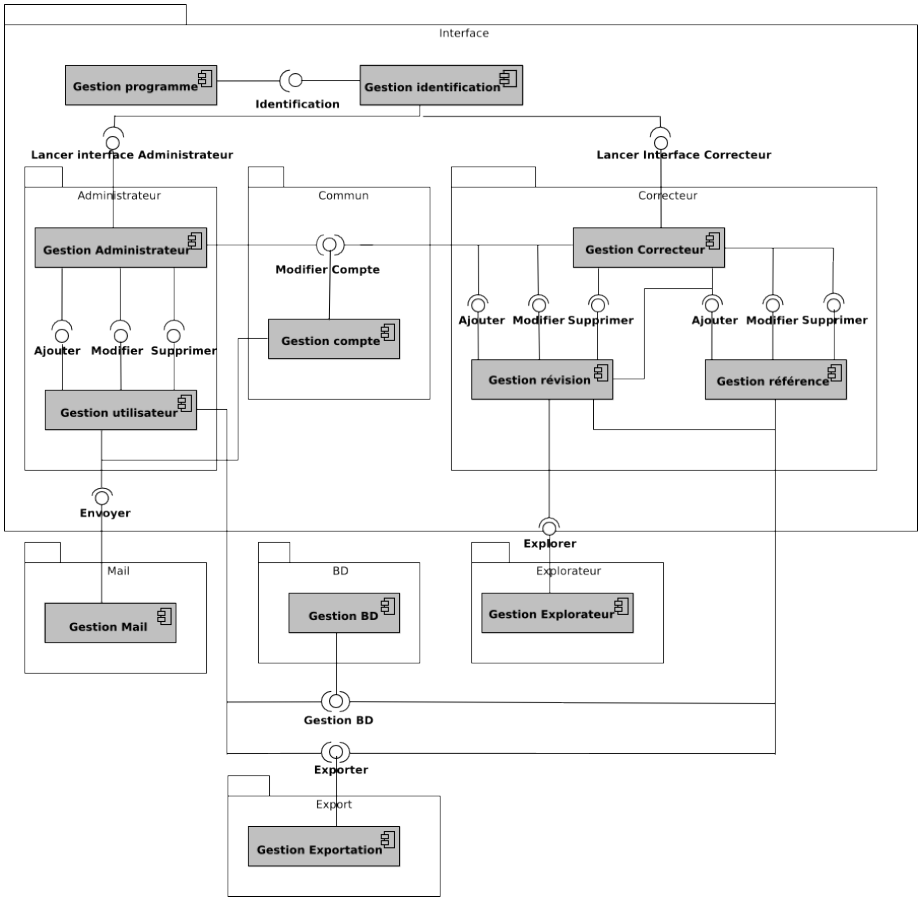
l'insertion des références lors de l'enregistrement d'une phase de révision de code via un arbre hiérarchique représentant les packages du projet par des nœuds et les références du projet par des feuilles.

Après avoir répercuté les commentaires des dirigeants de l'organisation, une deuxième phase de validation des exigences a été entreprise. Celle-ci était nécessaire afin de vérifier que nous avions bien pris en compte les modifications de la première phase. Une fois cette étape réalisée, nous avons pu passer à l'implémentation proprement dite du programme et à l'élaboration de la base de données.

# Implémentation

## Diagramme de composants du système

Le diagramme de composants décrit l'organisation du système du point de vue des modules de code. Ce diagramme permet de mettre en évidence les dépendances entre les composants (qui utilise quoi) et permet ainsi de mieux organiser les modules. L'encapsulation (insérer un composant dans un autre, cacher des détails d'implémentation,...) permet de réduire la complexité et d'élever le niveau d'abstraction. Le système développé pour Club Capra peut donc être représenté par le diagramme de composants suivant :



Nous allons maintenant décrire les différents composants du système ainsi que les fonctions qui leur sont attribuées.

* Gestion Programme

Ce premier composant a pour fonction de lancer le gestionnaire de revue de code. Une fois sa tâche terminée, il passe la main au gestionnaire d'identification.

* Gestion Identification

Ce gestionnaire a pour fonction la gestion d'identification des différents types d'utilisateurs, à savoir l'administrateur et le correcteur. Une fois l'administrateur identifié, il passe la main au gestionnaire de l'administrateur et dans le cas du correcteur, au gestionnaire du correcteur.

* Gestion Administrateur

Ce gestionnaire gère l'interface de l'administrateur. Il passera la main au gestionnaire de compte si l'administrateur invoque une fonctionnalité relative à la gestion de son compte ou au gestionnaire d'utilisateurs si l'utilisateur invoque une fonctionnalité relative à la gestion des utilisateurs.

* Gestion Correcteur

Ce gestionnaire gère l'interface du correcteur. Il passera la main au gestionnaire de compte si le correcteur invoque une fonctionnalité relative à la gestion de son compte, au gestionnaire de révisions si le correcteur invoque une fonctionnalité relative à la gestion des révisions ou au gestionnaire de références si le correcteur invoque une fonctionnalité relative à la gestion des références.

* Gestion Utilisateur

Ce gestionnaire gère les fonctionnalités relatives à la gestion des utilisateurs, à savoir : ajouter, modifier, supprimer, lister et exporter des utilisateurs. Pour leur bon fonctionnement, ce gestionnaire peut faire appel au gestionnaire

- de la base de données pour manipuler des données existantes ou en insérer de nouvelles,
- d'exportation afin d'exporter au format PDF certaines caractéristiques des utilisateurs enregistrés dans le système,
- de gestion des mails pour envoyer des informations aux utilisateurs si leur compte a été créé, modifié ou supprimé.

* Gestion Compte

Ce gestionnaire gère les fonctionnalités relatives au compte utilisateur, à savoir modifier son compte, passer en mode administrateur/correcteur, se déconnecter ou quitter le programme.

Pour les mêmes raisons que le composant "Gestion Utilisateur", il peut faire appel aux gestionnaires de la base de données et de gestion des mails.

* Gestion Révision

Ce gestionnaire gère les fonctionnalités relatives aux phases de révision de code, à savoir : ajouter, modifier, supprimer, lister ou exporter des révisions. Pour leur bon fonctionnement, ce gestionnaire peut faire appel au gestionnaire

- de la base de données pour manipuler des données existantes ou en insérer de nouvelles,
- de l'explorateur pour parcourir le projet associé à la phase de révision de code et y sélectionner les références à y inclure,
- des références pour enregistrer si nécessaire les références non encore présentes dans la base de données,
- d'exportation afin d'exporter au format pdf certaines caractéristiques des phases de révision de code enregistrées dans le système.

* Gestion Référence

Ce gestionnaire gère les fonctionnalités relatives aux références, à savoir : ajouter, modifier, supprimer, lister ou exporter des références. Pour leur bon fonctionnement, ce gestionnaire peut faire appel au gestionnaire de la base de données pour manipuler des données existantes ou en insérer de nouvelles et d'exportation afin d'exporter au format pdf certaines caractéristiques des références enregistrées dans le système.

* Gestion Mail

Ce gestionnaire s'occupe de l'envoi par e-mail des informations qui lui sont procurées par le gestionnaire appelant. Le mail est envoyé à l'utilisateur spécifié par ce même gestionnaire appelant.

* Gestion BD

Ce gestionnaire s'occupe de la base de données. Il regroupe les fonctions d'accès à la base de données et de gestion de son contenu (ajouter des données, récupérer des données, modifier des données,...).

* Gestion Export

Ce gestionnaire s'occupe d'exporter les données demandées par le gestionnaire appelant dans un rapport au format pdf selon certaines caractéristiques fournies par ce même gestionnaire appelant.

* Gestion Explorateur

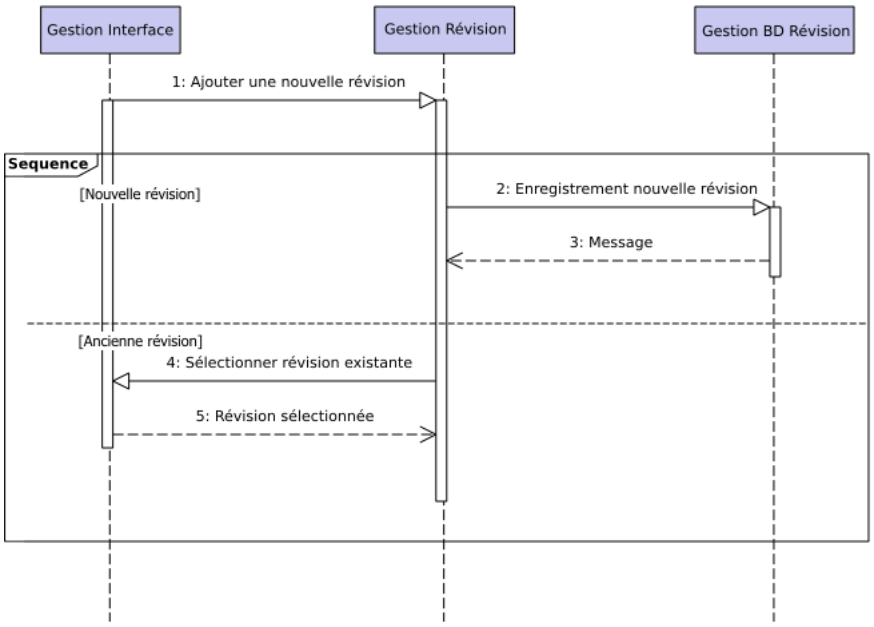
Ce gestionnaire offre au gestionnaire appelant un explorateur de fichiers selon un chemin déterminé sous forme d'un arbre comprenant des répertoires et des fichiers. Les nœuds représentent les packages du projet et les feuilles, les fichiers.

## Enregistrement d'une phase de révision

L'enregistrement d'une nouvelle phase de révision de code étant la fonctionnalité principale du système mais aussi la plus difficile à implémenter, elle a été représentée à l'aide de diagrammes de séquence. Ce processus d'enregistrement peut donc être divisé en 3 phases ; l'enregistrement ou la sélection de la phase de révision de code, l'enregistrement des références, si nécessaire, et l'enregistrement des erreurs détectées.

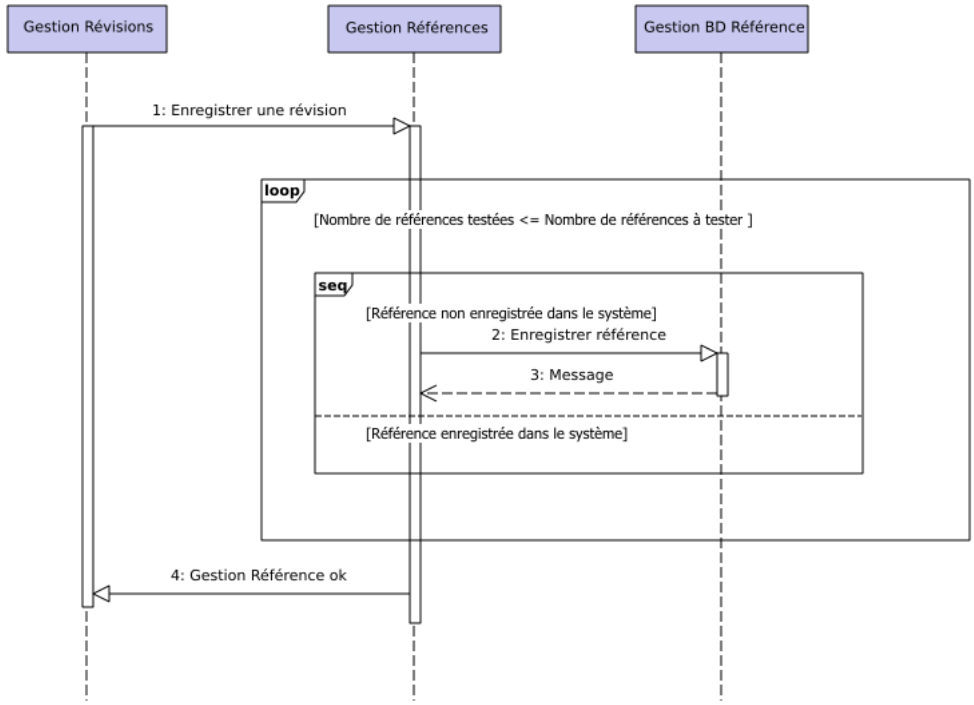
### Gestion de la révision

La première étape consiste à indiquer si les erreurs détectées lors de la phase de révision sont à ajouter à une ancienne phase ou à une nouvelle. Dans le premier cas, le gestionnaire d'interface reprend la main afin que l'utilisateur choisisse une phase de révision existante. Ensuite, il va renvoyer ce choix dans lequel les erreurs devront être répertoriées. Dans le deuxième cas, une nouvelle révision est ajoutée au système[1] .



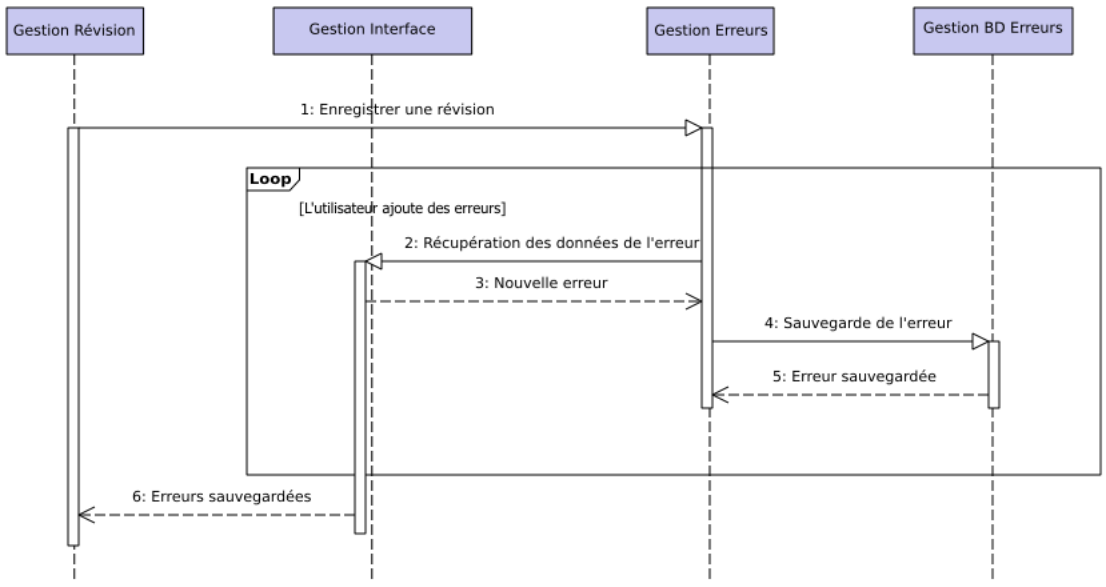

[1] Afin de rendre le schéma plus compréhensible, certains choix ont été entrepris :
– Gestion Interface est une généralisation de Gestion Administrateur et Gestion Correcteur du diagramme de composants et
– Gestion BD Révision est une spécification de Gestion BD du diagramme de composants.

### Gestion des références




Une fois la gestion de la révision effectuée, on passe à la gestion des références. En effet, il est possible que certaines références n'aient pas encore été enregistrées dans le système alors qu'elles sont inclues dans la phase de révision de code en cours d'enregistrement. C'est pourquoi, le gestionnaire de références va tester, pour chaque référence, si celle-ci est déjà enregistrée dans le système ou non. Si elle ne l'est pas encore, elle y sera automatique enregistrée[2].

### Gestion des erreurs




[2]Gestion BD Référence est une spécification de Gestion BD du diagramme de composants.

La dernière étape consiste à enregistrer les erreurs détectées lors de la phase de révision de code. C'est pourquoi, tant que l'utilisateur ajoute des erreurs via son interface, le gestionnaire d'erreurs va récupérer les éléments spécifiés par l'utilisateur et ajouter la nouvelle erreur dans le système[3].

Une fois que l'utilisateur spécifiera que toutes les erreurs ont été ajoutées, le gestionnaire de révisions reprendra la main, ce qui signifiera que le processus d'enregistrement de la phase de révision de code est terminée.

## Vérification de la solution

Une fois la phase d'implémentation terminée, nous sommes rentrés dans une phase de test nous permettant ainsi de vérifier notre programme sur base des différents scénarii établis lors de l'analyse des besoins. Ces tests ont été réalisés par nous-mêmes.

Une fois cette phase réalisée, le programme a été de nouveau testé mais cette fois-ci par deux personnes externes au projet : un doctorant de l'ETS travaillant en Génie logiciel (Zarour Mohammad) et un étudiant en dernière année à l'Institut d'Informatique de Namur (Pirmez Nicolas). Grâce à ces deux personnes externes, nous avons pu corriger certaines erreurs que nous n'aurions trouvées par nous-même puisque nous n'avions pas de recul vis-à-vis de notre travail. De plus, certains aspects relatifs à la manipulation du programme ont été suggérés par ces deux personnes et ont pu ainsi être insérés dans le programme.

[3] Afin de rendre le schéma plus compréhensible, certains choix ont été entrepris :
- Gestion Interface est une généralisation de Gestion Administrateur et Gestion Correcteur du diagramme de composants,
- Gestion BD Erreurs est une spécification de Gestion BD du diagramme de composants et
- Gestion Erreurs est une spécialisation de Gestion Révision du diagramme de composants.

# Annexe B :

*Modex - Implémentation*

# Implémentation de ModeX

> **Contenu :** *Description de l'implémentation, de la vérification et de validation de l'outil destiné à Horizon Vert : modeX.*

## Choix d'implémentation

L'implémentation du gestionnaire de requête de changement modeX s'est déroulée après celle du gestionnaire de revue de code réalisé pour Club Capra. ModeX a donc été basé sur les mêmes choix d'implémentation que ceux utilisés lors de cette première conception. En effet, ces choix se sont avérés performants et excellents dans leur utilisation. Le langage de programmation utilisé est donc le java via le programme Eclipse et la base de données est en MySql. L'exportation des rapports se fait toujours au format pdf.

La phase d'implémentation et de test ne se déroulant que sur un mois, une bonne partie du gestionnaire provient d'une adaptation de celui réalisé pour Club Capra. En effet, la gestion des mails, des utilisateurs, des comptes,... sont identiques.

Notre méthode d'implémentation pour élaborer le gestionnaire de revue de code est décrite ci-dessous, en commençant par la représentation du système et ensuite une explication brève de la phase de test.

## Validation des exigences

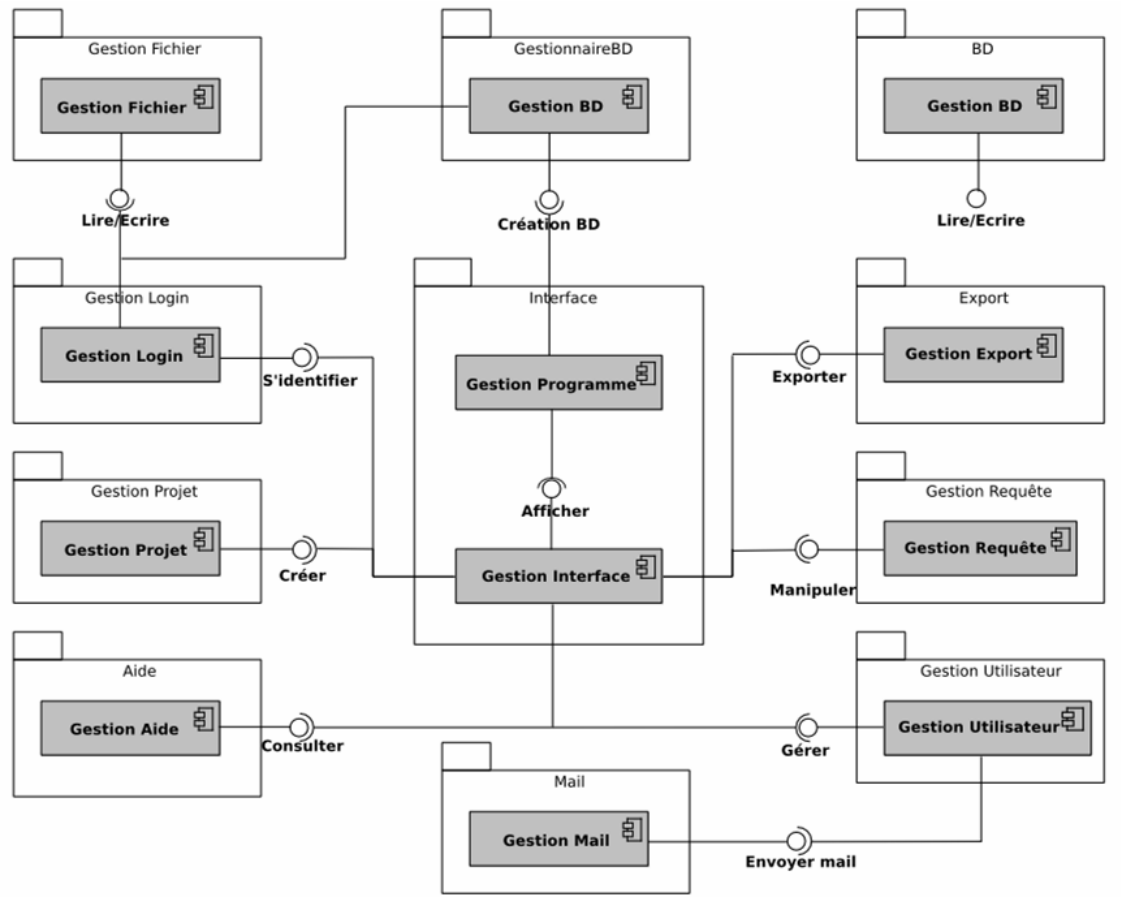
Une fois les choix d'implémentation réalisés, et dans le but de valider les exigences du client, nous avons réalisé des interfaces simples représentant graphiquement les différentes fonctionnalités du programme. En effet, cette méthode de validation des exigences s'est montrée très intéressante lors de l'implémentation du gestionnaire de révision de code pour Club Capra.

Les dirigeants d'Horizon Vert ont donc pu exprimer leur avis sur la manière dont le programma pouvait être réalisé. Ainsi, certaines idées et éléments ont pu être rajoutés au système. Par exemple, les dirigeants d'Horizon Vert ont demandé d'ajouter certains champs dans le template tels que le champ contenant la personne responsable d'implémenter le changement.

Une fois cette étape réalisée, l'implémentation proprement dite du programme et l'élaboration de la base de données nécessaire pour son fonctionnement ont pu être commencées.

## Implémentation

Le diagramme de composants décrit l'organisation du système du point de vue des modules de code. Ce diagramme permet de mettre en évidence les dépendances entre les composants (qui utilise quoi) et ainsi permet de mieux organiser les modules. L'encapsulation (insérer un composant dans un autre, cacher des détails d'implémentation,...) permet de réduire la complexité et d'élever le niveau d'abstraction. Le système développé pour Horizon Vert peut donc être représenté par le diagramme de composants suivant :




Le digramme de composants représenté par la figure VI peut être décrit en 3 parties :

## Gestion du programme

Le composant "Gestion Programme" a pour fonction de lancer le gestionnaire de requêtes de changement. Une fois sa tâche terminée, il passe la main soit au composant "GestionnaireBD.Gestion BD" s'il ne trouve pas les informations nécessaires pour se connecter à la base de données, ou au composant "Gestion Interface" dans le cas contraire.

## Gestion de la base de données

Lors du premier lancement du gestionnaire, il est nécessaire de passer par une phase d'installation permettant d'obtenir les informations nécessaires pour se connecter à la base de données, d'y insérer les tables si elles n'y existent pas encore et de créer son propre compte pour s'identifier au gestionnaire. Les composants nécessaires à cette première phase sont :

- **GestionnaireBD.GestionBD :** Ce composant contient l'ensemble des fonctionnalités nécessaires pour mener à bien la phase d'installation du sytème.
- **Gestion Fichier :** Il permet de sauvegarder les informations nécessaires pour se connecter à la base de données lors de chaque lancement du gestionnaire. Ce composant offre des méthodes permettant d'écrire et de lire dans un fichier créé préalablement par le gestionnaire.

Une fois la phase d'installation terminée, la main est rendue au composant "Gestion Programme" qui va ensuite appeler le composant "Gestion Interface" pour afficher le gestionnaire de requêtes de changement.

## Gestion des interactions

Le composant "Gestion Interface" constitue le composant principal du système. En effet, selon l'interaction de l'utilisateur, il va passer la main à un autre composant qui va réaliser la tâche demandée et ensuite la lui redonner. Tous ces composants interagissent avec la base de données via le composant "BD"[4]. Plusieurs cas sont possibles :

* Gestion Login

Ce composant s'occupe de l'identification de l'utilisateur lors du lancement de modeX et de la récupération du mot de passe. Il interagit avec le composant "Gestion Fichier" afin de récupérer les informations nécessaires pour se connecter à la base de données et ainsi, vérifier si l'utilisateur est bien la personne qu'il prétend être[5].

* Gestion Projet

Ce composant s'occupe de créer les projets que le chef de projet spécifie. Une arborescence est alors créée pour chaque projet afin d'y enregistrer les rapports d'exportation.

* Gestion Aide

Ce composant s'occupe d'afficher à l'utilisateur le fichier d'aide livré avec le gestionnaire de requêtes de changement modeX.

[4] Aucun lien n'a été tracé vers ce composant afin de ne pas alourdir le diagramme

[5] Le fichier contient ces informations puisqu'une phase d'installation a été réalisée lors du premier lancement de modeX.

* Gestion Export

Ce composant s'occupe d'exporter le rapport d'un projet déterminé selon les différentes caractéristiques fournies par l'utilisateur (Parties à intégrer, ordre de classement,...).

* Gestion Requête

Ce composant contient les fonctions d'ajout d'une nouvelle requête, de modification et de suppression d'une requête déjà existante ainsi que de l'affichage des détails et exigences dépendantes d'une requête sélectionnée préalablement.

* Gestion Utilisateur

Ce composant gère les fonctionnalités relatives à la gestion des utilisateurs, à savoir : ajouter et supprimer un utilisateur. Il travaille avec le composant "Gestion Mail" afin d'avertir l'utilisateur de la création ou de la suppression de son compte.

## Vérification de la validation

Une fois la phase d'implémentation terminée, nous sommes rentrés dans une phase de test permettant ainsi de vérifier le programme sur base des différents scénarii établis lors de l'analyse des besoins. Cette phase de test a duré une semaine et a permis de livrer un gestionnaire fonctionnant correctement dans les situations normales et étant le plus stable dans les situation anormales.

# Annexe C :

*Version actuelle de la micro-évaluation*

# Version actuelle du questionnaire

> **Contenu :** *Présentation de la structure actuelle du questionnaire de la micro-évaluation.*

## Structure du questionnaire

La structure actuelle du questionnaire de la micro-évaluation est basée sur 6 axes, tels que présenté comme suit :

### Gestion de la qualité

- *Implication pour la qualité :*
  1. Quelles règles et/ou procédures systématiques ont été utilisées dans le cadre de ce projet ?
- *Origine de la qualité :*
  2. Quels moyens mettez-vous en oeuvre pour garantir que ce qui a été développé corresponde aux exigences du client ?

### Gestion des clients

- *Formalisation des exigences :*
  3. Comment sont récoltées et formalisées les exigences du projet ?
- *Gestion des changements :*
  4. Les exigences originales peuvent-elles être revues, peuvent-elles évoluer en cours de projet ?
- *Intégration du client :*
  5. Quel rôle le client joue-t-il en cours de projet ?

### Gestion des sous-traitants

- *Formalisation des exigences à sous-traiter :*
  6. Comment formalisez-vous les exigences que vous destinez à vos sous-traitants ?
- *Coordination avec les sous-traitants :*
  7. Quelles relations, quels contacts avez-vous avec vos sous-traitants pendant le déroulement de leur intervention sur le projet ?

### Développement et gestion de projet

- *Découpage du projet :*
  8. Comment était organisé le projet, était-il découpé en phases/étapes définies ?
- *Planification :*
  9. Le projet a-t-il fait l'objet d'un planning ?
- *Suivi des ressources :*
  10. Pouvez-vous connaître à tout moment les ressources consommées sur le projet ?
- *Suivi de l'avancement :*
  11. Pouvez-vous connaître à tout moment l'état d'avancement du projet ?
- *Collaboration au sein des équipes :*
  12. Comment les membres de l'équipe collaborent-ils ensemble en cours de projet ?
- *Techniques de développement :*
  13. Comment faites-vous pour détecter et corriger les erreurs en cours de projet ?

### Gestion de produit

- *Gestion des versions :*
  14. Comment gérez-vous les différentes versions des modules développés et des documents produits dans le cadre de la gestion de projet ?
- *Standardisation :*
  15. Utilisez-vous des documents type, des structures pré-définies pour les produits de votre travail ? Des règles de programmation sont-elles suivies ?

### Gestion des compétences

- *Gestion des compétences :*
  16. Les membres de l'équipe avaient-ils les compétences nécessaires pour assurer les tâches qui leur ont été confiées ?

# Annexe D :

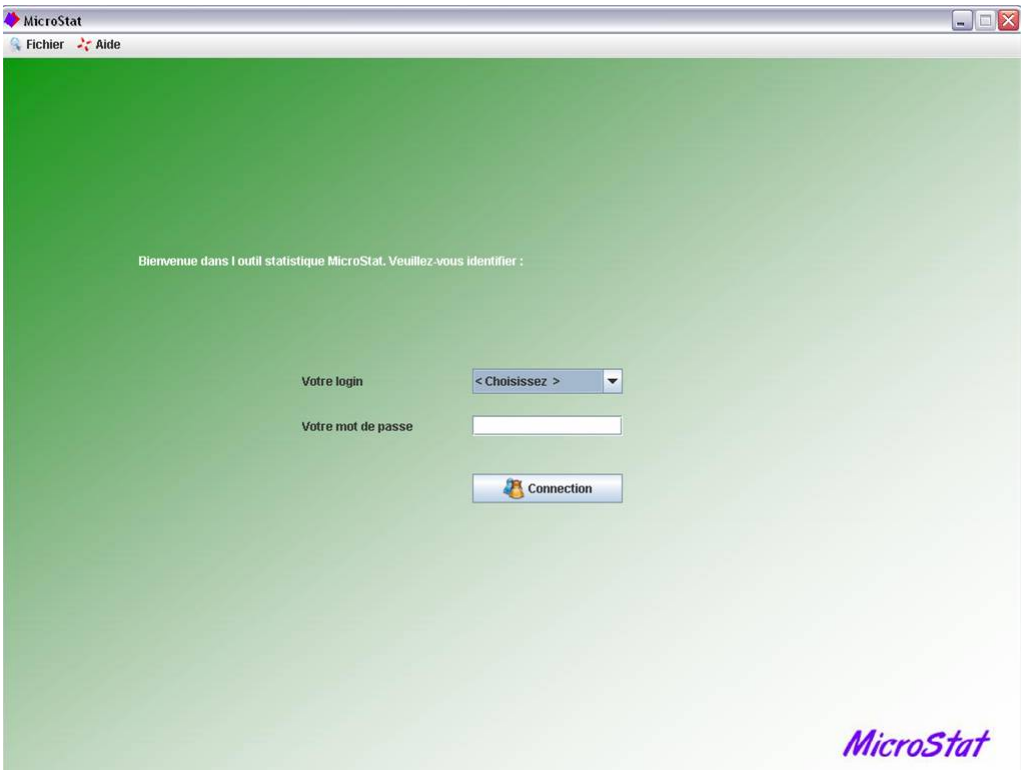
*Manuel d'aide de MicroStat*

# Identification

> **Contenu :** *Aide nécessaire à l'identification d'un utilisateur pour rentrer dans MicroStat.*

## Identification

Lors du démarrage de MicroStat, la première fenêtre qui s'affiche permet de s'identifier dans le système. Il faudra donc y posséder un compte, à savoir, un login et un mot de passe.




Le login est composé d'un numéro d'identification unique, généré automatiquement lors de la création du compte, ainsi que du nom de la personne. Le mot de passe doit, quant à lui, être composé de minimum 5 et maximum 10 caractères.

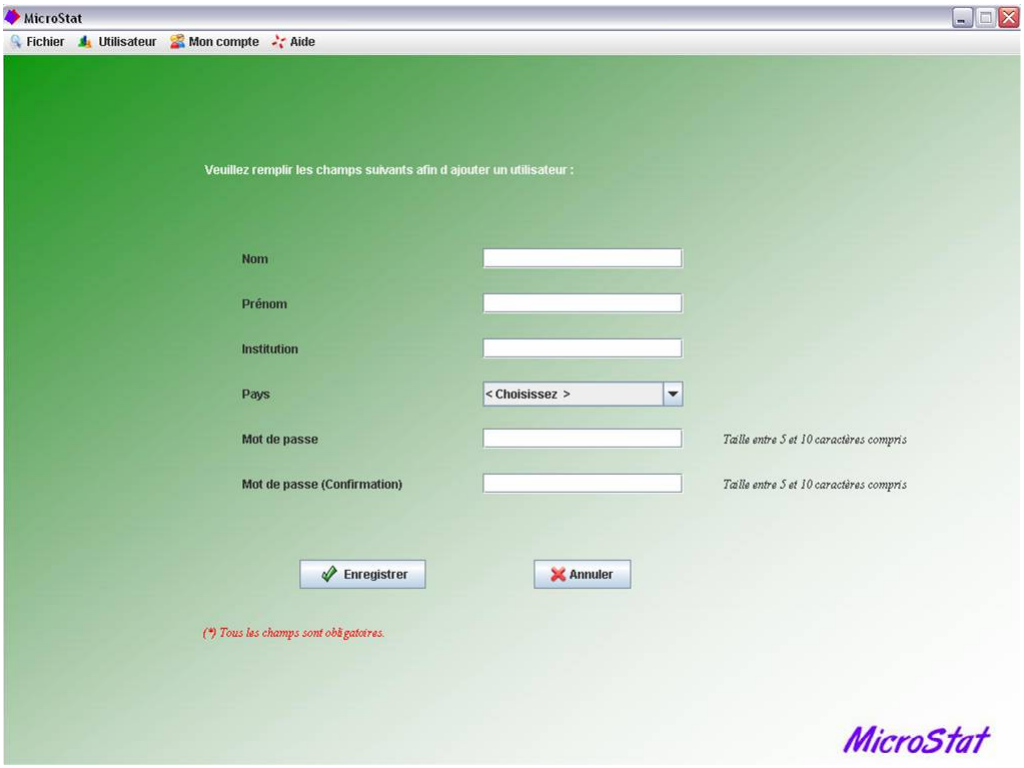
Suivant le type de compte - « Administrateur » ou « Utilisateur » - l'utilisateur pourra accéder ou non à certaines fonctionnalités de MicroStat. Ci-après, les fonctionnalités disponibles pour chacun des deux types de compte.

# Gestion des utilisateurs

> **Contenu :** *Explication de l'ajout, de la suppression d'un utilisateur ainsi que la modification de d'un compte utilisateur.*

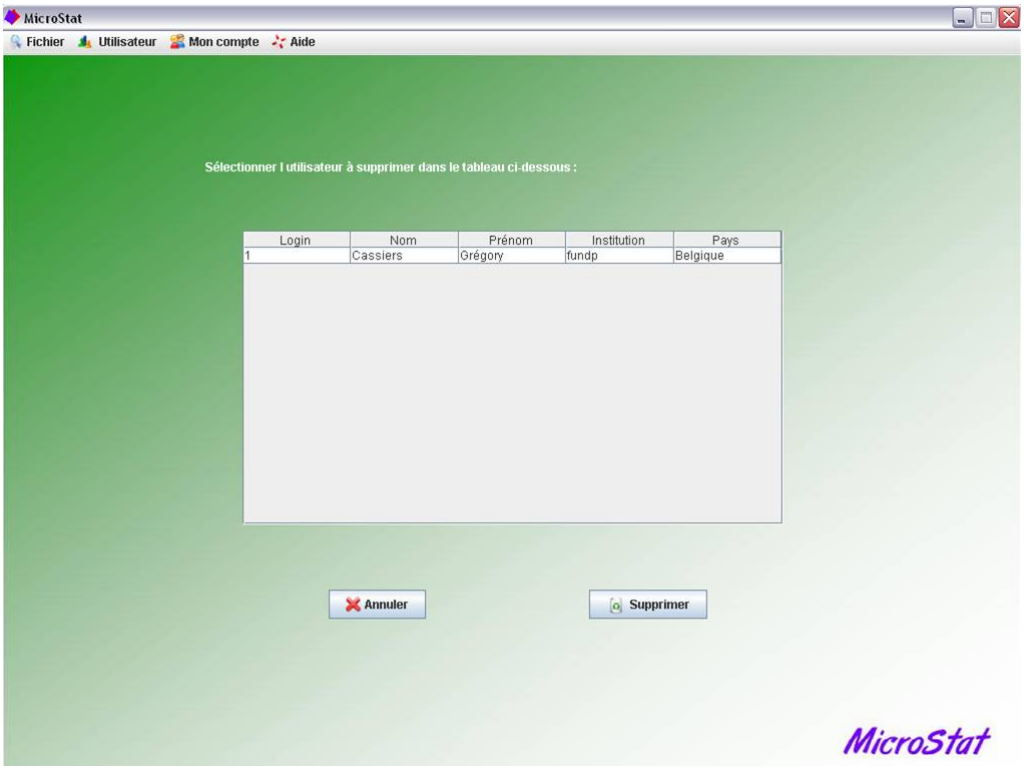
## Ajout d'un utilisateur

Seul l'administrateur peut accéder à la fonctionnalité relative à l'ajout d'un nouvel utilisateur. Lors de l'ajout, l'administrateur devra spécifier différentes informations telles que celles de la figure suivante :



## Suppression d'un utilisateur

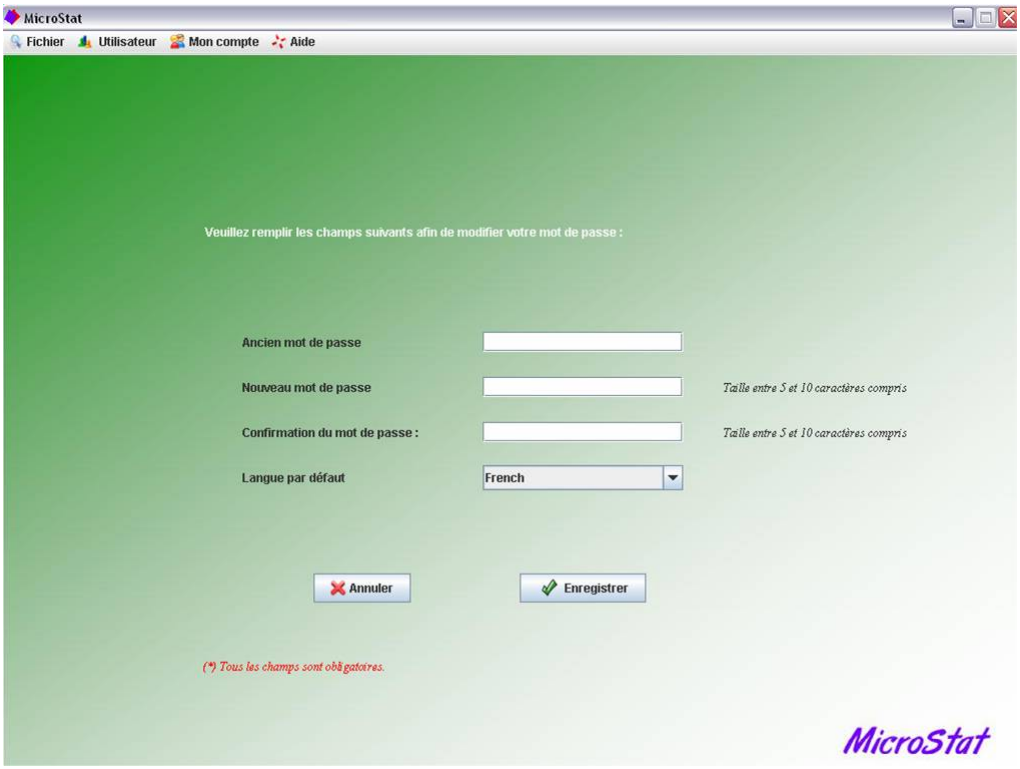
Seul l'administrateur peut accéder à la fonctionnalité relative à la suppression d'un utilisateur. Pour supprimer un utilisateur, l'administrateur doit sélectionner l'utilisateur dans la liste et ensuite le supprimer.




Il n'est pas possible de supprimer plusieurs utilisateurs en même temps.

## Modification de son compte

La modification de son propre compte est une fonctionnalité accessible à l'administrateur et à l'utilisateur. Cette fonctionnalité permet de modifier son mot de passe ainsi que la langue du programme.




Le changement de la langue du programme prend effet directement une fois que l'administrateur ou l'utilisateur a appliqué ce changement.

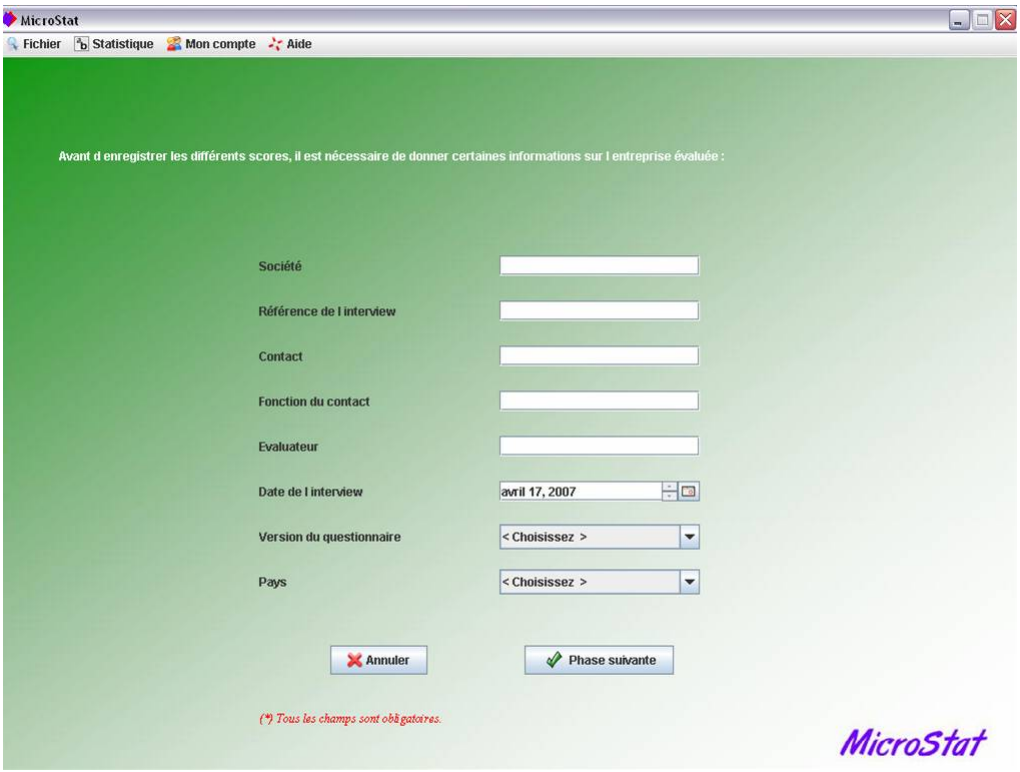
# Gestion des questionnaires

> **Contenu :** *Aide nécessaire à la gestion des questionnaires des entreprises évaluées via la micro-évaluation.*

## Enregistrer un questionnaire

L'enregistrement d'un questionnaire se compose de deux phases : l'enregistrement des informations sur l'entreprise évaluée et l'enregistrement des scores relatifs à cette entreprise.

## Enregistrer les informations sur l'entreprise

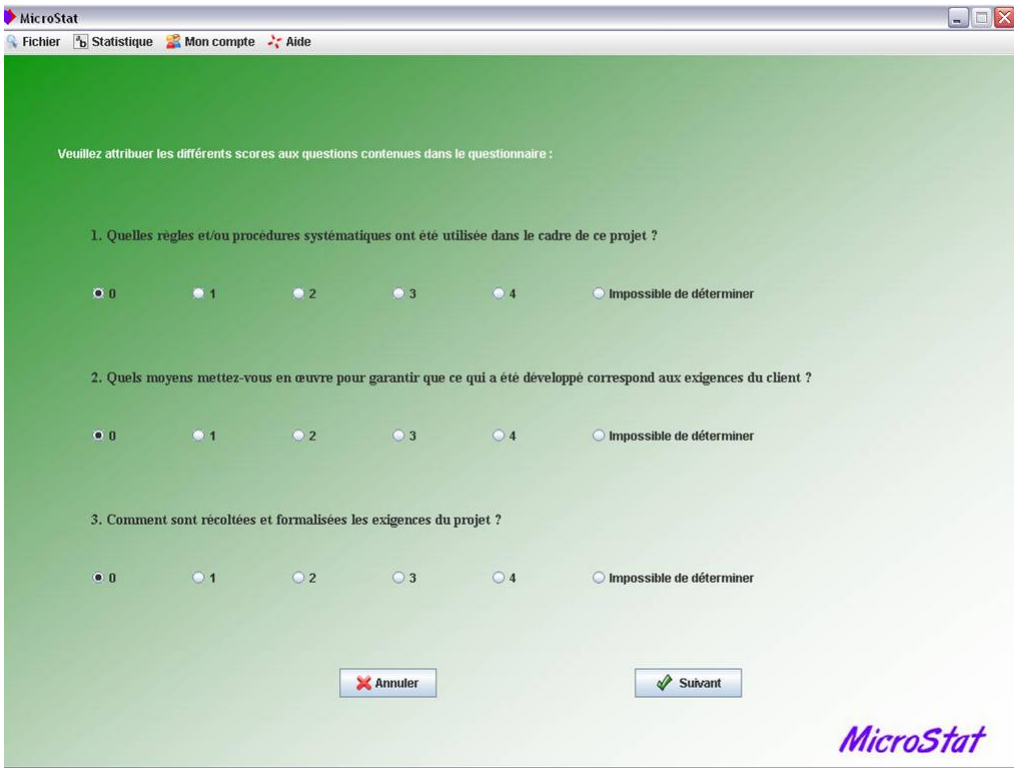



Cette première étape consiste à enregistrer les informations relatives à l'entreprise évaluée. Les informations à spécifier sont celles reprises dans la figure ci-dessus. La référence de l'interview doit être spécifiée comme suit :

Une fois cette étape réalisée, il reste à enregistrer les scores que l'entreprise a obtenus suite à l'application de la micro-évaluation.

## Enregistrer les scores de l'entreprise

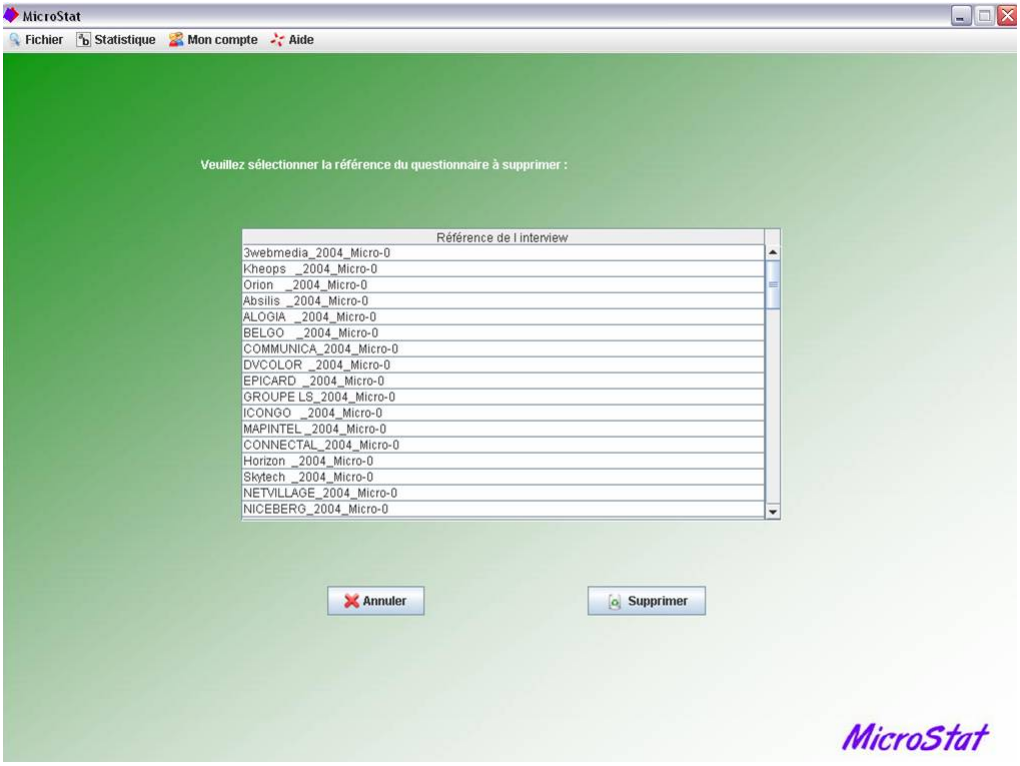
Cette deuxième étape consiste à spécifier, pour chaque question, le score obtenu par l'entreprise. Si aucun score n'a pu être déterminé, il faut spécifier la valeur "Impossible à déterminer" comme score obtenu.




Pour chaque question, le score par défaut qui est spécifié équivaut à zéro. Il faut donc faire attention à bien avoir spécifié les scores obtenus par l'entreprise lors de l'enregistrement de celle-ci.

## Supprimer un questionnaire

Pour supprimer un questionnaire, l'utilisateur doit sélectionner la référence de l'interview correspondant au questionnaire dans la liste et ensuite le supprimer.



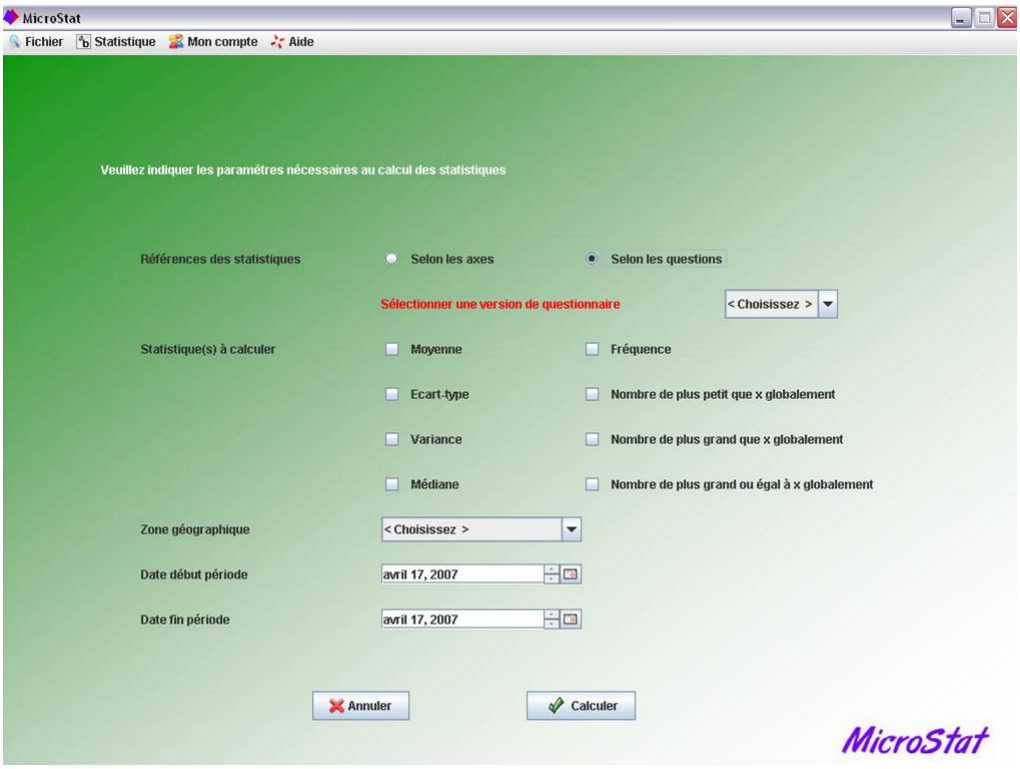

Il n'est pas possible de supprimer plusieurs questionnaires en même temps.

# Calculer des statistiques

**Contenu :** *Aide nécessaire au calcul des divers statistiques qu'offre MicroStat.*

## Calculer les statistiques

Afin de calculer des statistiques via MicroStat, il est nécessaire de spécifier certains critères tels que repris dans la figure ci-dessous :




Il faut d'abord spécifier la référence des statistiques. Celle-ci sera établie soit selon les axes si vous désirez que les versions des questionnaires ne soient pas prises en compte ou soit selon les questions dans le cas inverse. C'est pourquoi, lors du choix d'un référencement par questions, il est nécessaire de sélectionner la version du questionnaire sur laquelle les statistiques devront être élaborées. Il reste ensuite à spécifier les statistiques à calculer, la zone géographique et les dates de début et de fin de période de votre échantillon.

Une fois les statistiques élaborées, l'outil vous proposera de sauvegarder le fichier Excel contenant l'échantillon et les calculs.

## Calculs statistiques possibles

MicroStat permet de calculer les statistiques suivantes[6] :

### Moyenne

On calcule la moyenne d'une variable numérique en additionnant les valeurs de toutes les observations incluses dans un ensemble de données, puis en divisant cette somme par le nombre d'observations qui font partie de l'ensemble. Ce calcul permet d'obtenir la valeur moyenne de toutes les données.

### Médiane

Si les observations d'une variable sont ordonnées par valeur, la valeur médiane correspond à l'observation qui se trouve au point milieu de cette liste ordonnée. Elle correspond plus précisément à un pourcentage cumulé de 50% (c'est-à-dire que 50% des valeurs sont supérieures à la médiane et 50% lui sont inférieures). La position de la médiane est donc la valeur (n+1)/2, le **n** désignant le nombre de valeurs dans un ensemble de données.

### Écart-type

L'écart-type est la mesure de dispersion la plus couramment utilisée en statistique lorsqu'on emploie la moyenne pour calculer une tendance centrale. Il mesure donc la dispersion autour de la moyenne En raison de ses liens étroits avec la moyenne, l'écart-type peut être grandement influencé si cette dernière donne une mauvaise mesure de tendance centrale.

L'écart-type est aussi influencé par les valeurs aberrantes ; une seule de ces valeurs pourrait avoir une grande influence sur les résultats de l'écart-type. Il s'agit donc d'un bon indicateur de l'existence de valeurs aberrantes, ce qui en fait une mesure de dispersion très utile pour les distributions symétriques ne comptant aucune valeur aberrante, tel que dans la micro-évaluation.

### Variance

La variance est une mesure du degré de dispersion d'un ensemble de données. On la calcule sous la forme de l'écart au carré moyen de chaque nombre par rapport à la moyenne d'un ensemble de données.

[6] Les définitions proviennent de http ://www.statcan.ca/francais/

### Fréquence

La fréquence d'une observation particulière est le nombre de fois que l'observation se dégage des données. La distribution d'une variable est le profil des valeurs de l'observation. Les distributions de fréquences peuvent être représentées sous forme de tableaux ou de diagrammes.

### Nombre de résultat suivant une condition x globalement

Cette statistique détermine le nombre d'éléments contenus dans l'échantillon sélectionné vérifiant la condition x. Dans le cas de l'outil statistique MicroStat, la condition x peut être de trois types différents, ce qui donne trois calculs statistiques possibles :

1. Nombre de résultats plus petits que x, x $\in$ {1,2,3,4}
2. Nombre de résultats plus grands que x, x $\in$ {0,1,2,3}
3. Nombre de résultats plus grands ou égaux à x, x $\in$ {1,2,3,4}

# Septième partie

# Rapports de stage

# Rapport A :

*Capra - Plan d'action*

# Avant-propos

Ce document consiste en une synthèse détaillée du plan d'action élaboré pour le Club Capra de l'Ecole de Technologie Supérieur (ETS). Ce rapport se base principalement sur le rapport de la micro-évaluation réalisé sur Capra le 20 mai 2006. Suite à cette première analyse, nous avons fait ressortir 5 problèmes clés, présentés de la priorité la plus haute à la plus basse :

1. **Gestion de la qualité - Revue de code :** La revue de code semble être le problème le plus important dans la gestion de la qualité chez Capra puisqu'il n'existe aucun outil pour cette technique.
2. **Gestion de la qualité - Tests unitaires :** Il n'existe pas de tests unitaires chez Capra. Les tests structurels pourraient être utiles comme complément à la revue de code tandis que les tests fonctionnels nous semblent trop lourds à implémenter pour un club d'étudiants.
3. **Gestion des changements :** Ce problème fait référence à du moyen terme et fait référence à la formalisation des exigences.
4. **Méthodologie de développement :** Club Capra n'utilise aucune méthodologie de développement. C'est un problème à moyen terme.
5. **Gestion des risques :** Capra ne possède aucune gestion des risques.

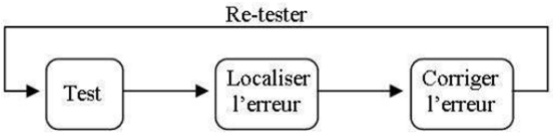
Ce que nous pouvons réaliser pour Capra doit dépendre du court terme puisqu'il nous est seulement possible de déployer une application sur 2 mois. Il semble donc évident que les problèmes de méthodologie de développement et de gestion des changements ne doivent pas être retenus. Quant à la gestion des risques, ce problème semble être d'une priorité mineure pour le Club.

Nous proposons donc pour le Club Capra de travailler sur la **gestion de sa qualité** et plus précisément sur sa **revue de code**. De même, l'élaboration de tests structurels semblent être un bon complément une fois la technique de revue de code implantée. Quant aux tests fonctionnels, ceux-ci nous semblent trop lourds et fastidieux à réaliser pour un club d'étudiants.

La suite du document reprend les différents problèmes décelés chez Capra ainsi qu'une possibilité de solution à ce problème. Ils sont présentés du plus prioritaire au moins prioritaire.

# Gestion de la qualité - Revue de code

## Revue de code




La revue de code permet de détecter un bon nombre d'erreurs. Cette phase de test est réalisée par une équipe différente de celle qui a implémentée le module (il est en effet difficile de corriger son propre code). Les erreurs doivent alors être rapportées à l'équipe de codage afin qu'elles soient corrigées si nécessaire. La technique de revue du code permet de détecter environ 85% des erreurs mais celles-ci ne sont que superficielles (mauvaise condition de boucle,...).

Dans le cadre du Club Capra, puisque la programmation se fait par module et que des équipes différentes travaillent sur chaque module, il est alors facile de mettre en place la technique de revue de code : une fois un module terminé, une autre équipe revoit le code et rapporte les erreurs à l'équipe qui a codé le module. Il suffirait alors de travailler avec un document type pour rapporter les erreurs.

**Numéro :** .....
**Localisation :**
*Nom du fichier :* .....
*Numéro de ligne :* .....
**Description :** .....
**Correcteur :** .....
**Date de la revue :** .....
**Date de la correction :** .....

L'avantage de cette technique est qu'elle n'est pas coûteuse et assez rapide. Une bonne équipe peut aller jusqu'à 500 instructions par heure. On pourrait alors créer un programme java qui permettrait d'encoder les différentes erreurs et qui donnerait en sortie un document reprenant toutes les erreurs encodées.

# Gestion de la qualité - Tests unitaires

Dans la réalisation de jeux de tests, il est nécessaire d'établir des classes d'équivalence telles que pour chaque classe, un test sur un membre de la classe suffit. Pour choisir ces classes d'équivalence, deux bases sont possibles :

- sur base de la structure du programme : tests structurels (glass-box)
- sur base de la spécification du programme : tests fonctionnels (black-box)

## Tests structurels - Glass box

Réaliser des tests structurels peut être considéré comme un complément à la revue du code. En effet, cette phase de test nécessite une analyse de l'algorithme afin de déterminer :

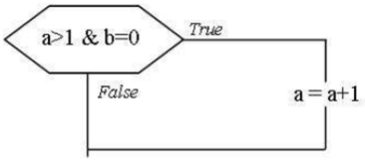
- Les différents points de décision.
- L'identification des différents chemins possibles.
- Le calcul du prédicat chemin c'est-à-dire définir les propriétés des inputs pour être sûr que le contrôle passe par tel ou tel chemin.

A partir de ces trois points, il est alors possible d'établir des valeurs de test pour chaque critère choisi. Ces différents critères sont :

- Couvrir chaque instruction
- Couvrir chaque issue de chaque condition,...

Un jeu de tests peut alors être établi de sorte qu'il couvre tout l'algorithme.

### Exemple :




- **Critère choisi :** Couvrir chaque issue de chaque point de décision
- **Idée :** Un test qui couvre la sortie true et un autre qui couvre la sortie false
- **Proposition de valeurs de test :** a=2 ; b=0 et a=3 ; b=1

## Tests fonctionnels - Black box

Dans le cadre d'élaboration de tests fonctionnels, il existe pour la langage C++ un ensemble de programmes permettant d'établir ce type de test. Il est nécessaire de se baser sur un critère ou une combinaison des critères :

1. **propriétés des données :** Il faut lister les propriétés significatives des données et établir des classes grâce à celles-ci. Ensuite, il faut trouver un jeu de tests qui recouvre le maximum de classes et un jeu de tests qui recouvre une et une seule classe.
2. **Cas limites des données :** L'idée est de choisir des représentants situés de part et d'autre de la frontière entre classes d'équivalences. C'est un bon complément à l'approche par propriétés des données puisque cela entraîne de choisir comme représentants de classe les cas limites et de tenir compte également des cas limites de classes de résultats (ex : premier et dernier élément d'une liste).
3. **Relations "cause à effet" :** L'idée est de se baser sur des combinaisons logiques des propriétés des inputs/outputs correspondants.

Sur base du critère choisi, il ne reste plus qu'a établir les propositions de valeurs de test afin de construire un jeu de tests qui couvre tout l'algorithme.

### Exemple :

Prenons le cas d'un algorithme qui permet de modifier la représentation d'un texte donné via un ensemble de commande que voici :

- **G :** Commande qui transforme le texte en gras
- **I :** Commande qui transforme le texte en italique
- **E :** Commande qui transforme le texte en Evidence c'est-à-dire en gras
- **SE :** Commande qui transforme le texte en Super Evidence c'est-à-dire en gras et italique

Le critère utilisé pour cette exemple est celui de relations "cause à effet" puisque suivant un input choisi (la commande), on obtient un output déterminé. On peut donc établir le jeu de tests fonctionnels suivant, en regroupant les inputs donnant le même output :

| # | *Input* | *Output* |
|---|---|---|
| 1 | G, E | Texte en gras |
| 2 | I | Texte en italique |
| 3 | SE | Texte en gras et italique |

# Gestion des changements

Une bonne gestion des changements implique deux points importants qui amènent un traçabilité dans le projet :

- **Gestion des exigences :** elles doivent être bien formalisées grâce à un document type.
- **Processus de développement :** méthodologie de développement adaptée à l'entreprise, qui utilise des cycles itératifs courts permettant l'intégration rapide de changements.

## Gestion des exigences

La description des exigences fonctionnelles et non fonctionnelles du système peut se réaliser via le template Volere qui offre une carte de description des exigences. On pourrait alors réaliser un programme qui permettrait à l'utilisateur de remplir ses fiches et qui donnerait un rapport reprenant l'ensemble des fiches entrées.

<table>
<tr><td>Exigence # : identifiant unique</td><td>Type d'exigence : référence au modèle</td><td>Événements : Liste d'événements qui nécessitent cette exigence</td></tr>
<tr><td colspan="3">Description : objectif de l'exigence en une phrase</td></tr>
<tr><td colspan="3">Justification : raison pour laquelle cette exigence est présente</td></tr>
<tr><td colspan="3">Origine : qui a émis cette exigence ?</td></tr>
<tr><td colspan="3">Critère de satisfaction : une mesure de l'exigence qui permet de tester si la solution proposée remplit l'exigence initiale.</td></tr>
<tr><td colspan="2">Contentement du maître d'ouvrage : degré de contentement du maître d'ouvrage si le produit final satisfait cette exigence. De 1 (pas intéressé) à 5 (très content).</td><td>Mécontentement du maître d'ouvrage : degré de mécontentement du maître d'ouvrage si le produit final ne satisfait pas cette exigence. De 1 (quasi indifférent) à 5 (très mécontent)</td></tr>
<tr><td colspan="2">Exigences dépendantes : liste d'exigences dont l'implémentation dépend de l'implémentation de celle-ci.</td><td>Exigences conflictuelles : exigences qui ne peuvent pas être implémentées si celle-ci l'est.</td></tr>
<tr><td colspan="3">Documents relatifs : référence à des documents qui illustrent et expliquent cette exigence.</td></tr>
<tr><td colspan="3">Historique : création, modifications, destruction apportées à l'exigence.</td></tr>
</table>

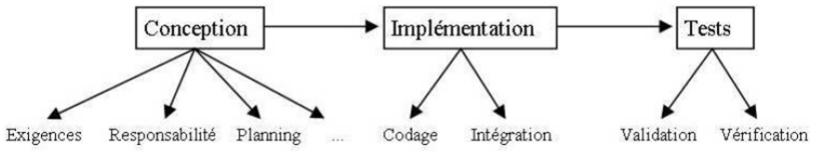
Numéroter les exigences sert à tracer les exigences au cours du processus de développement.

* Le numéro de l'exigence est un numéro unique, choisi de manière séquentielle.
* Le type d'exigence est le numéro de la rubrique du cahier des charges dans laquelle sera classé ce type d'exigence. Exemple : numéro 9 pour les exigences fonctionnelles.
* Les degrés de contentement et de mécontentement du maître d'ouvrage sont à demander au maître d'ouvrage.

# Méthodologie de développement

Pour le Club Capra, vu qu'il travaille par module, il me semblerait logique de travailler avec un cycle de développement itératif qui comporterait les étapes suivantes :

- Conception : cette phase consisterait à revoir les cartes des exigences, les responsabilités et le planning.
- Implémentation : codage des modules et intégration.
- Tests : Vérification et Validation c'est-à-dire revue de code et tests fonctionnels




Pourquoi utiliser une méthode de développement par itérations et non linéaire ou par prototype ? Quels en sont les avantages ?

+ chaque développement est moins complexe
  les intégrations sont progressives
+ permet la livraison par morceaux
+ permet un apprentissage progressif
+ réagit plus facilement aux changements
+ meilleure gestion de compétences

Cependant, cette méthode de développement nécessite une planification bien documentée, des exigences claires,... Si l'opportunité se présente, la technique de réutilisabilité peut être adoptée.

Une phase d'élaboration d'un cahier des charges pourrait aussi être utile afin de prioritiser les exigences, de définir les responsabilité et d'avoir une compréhension commune de ce qu'il faut réaliser et la manière de le réaliser. Plusieurs outils peuvent alors être utilisés pour améliorer le développement :

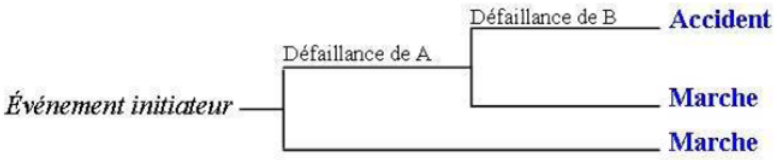
- **GanttProject :** pour établir un planning
- **Visio ou Visual Paradigm :** pour élaborer le cahier des charges

# Gestion de risque

La gestion des risques est concernée par :
- L'identification des risques qui peuvent affecter le projet,
- La planification pour réduire l'ampleur des risques pour qu'ils ne se développent pas en menaces principales.

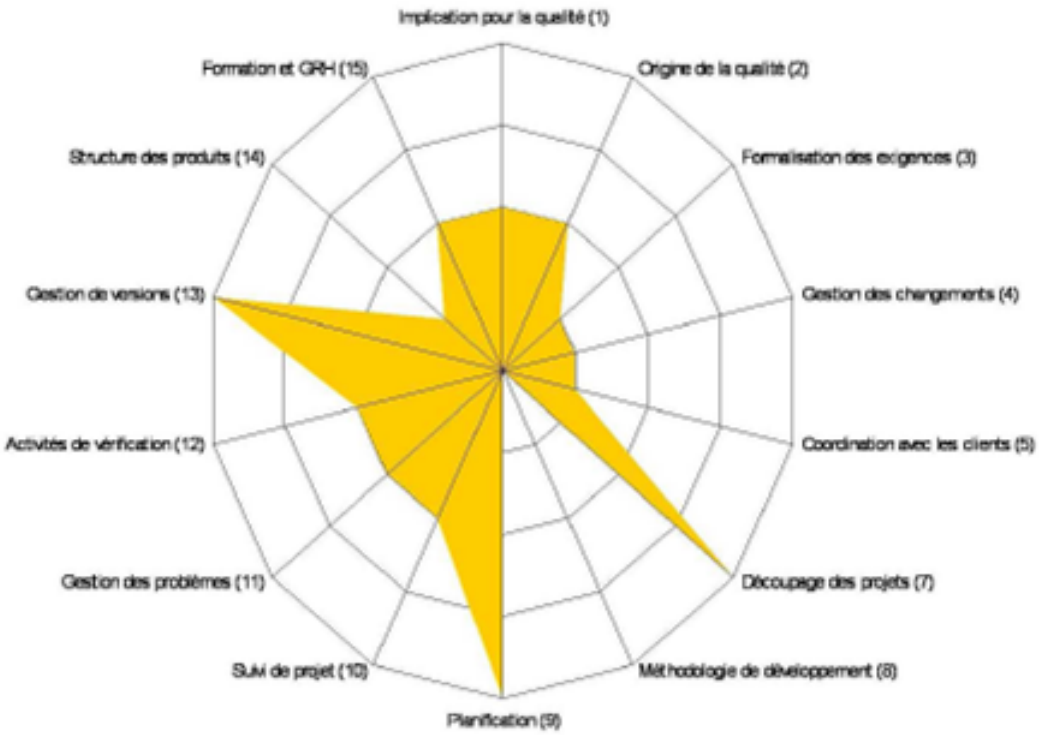
Il existe différents types de risques :
- **Les risques de projet :** Ils affectent l'horaire ou les ressources → *Capra*
- **Les risques de produit :** Ils affectent la qualité ou la performance du logiciel étant développé.
- **Les risques d'entreprise :** Ils affectent l'entreprise développant ou obtenant le logiciel.

Une méthode d'identification des risques qui peut être utilisée par le Club Capra est l'arbre d'événement. Il consiste en une description du scénario d'accident produit par un enchaînement de défaillances suite à l'occurrence d'un événement initiateur.



# Synoptique détaillée

Le graphique ci-dessous représente les résultats du questionnaire de la micro-évaluation pour les différents axes.



# Bibliographie


[1] **Micro-évaluation**, *Rapport de micro-évaluation des pratiques logiciels - Capra*, Mathieu Viau, Ba Tuan-Anh Pham, Damien Dubé.

[2] **OWPL**, *L'amélioration de la performance des processus et pratiques logiciels dans les petites entreprises françaises*, Anabel Strambolian, Avril 2006.

[3] **OWPL**, *Une Méthodologie et des Modèles Légers pour Initier une Démarche d'Amélioration des Pratiques Logicielles APL*, Naji Habra, Alain Renault.

[4] **Génie logciel**, *Ingénierie des logiciels*, Naji Habra, F.U.N.D.P.

[5] **Ingénierie des exigences**, *Requirements engineering*, Patrick Heymans, F.U.N.D.P.

[6] **Volere**, *Requirements Specification Template*, Edition 10.1, Atlantic Systems Guild.

[7] **Gestion des risques**, *Gestion de projets logiciels*, Tom Mens, UMH.

[8] **Gestion des risques**, *La gestion des risques*, Ledru Chrystophe, Plantec Mickaël, Scanff Arnaud, ISTIA.

# Rapport B :

*Capra - Cahier des charges*

# Description des classes d'utilisateurs

## Administrateur

## Caractéristiques

1. **Attributs physiques :** La personne à l'âge légal pour exercer la fonction d'administrateur. Il n'y a à priori aucun âge spécifique ni de sexe typique. Rien n'est prévu au sein du logiciel pour la prise en charge d'utilisateurs à capacités réduites. Des incapacités physiques et handicaps modérés sont cependant tolérés.
2. **Attributs mentaux :** L'administrateur est familiarisé aux concepts modernes des interfaces graphiques. Notre application n'est pas adaptée aux handicaps lourds. Du fait de sa qualification et de son intérêt évident pour la revue de code, l'administrateur adopte une attitude positive vis-à-vis de la tâche.
3. **Qualifications et connaissances :** La langue maternelle de l'administrateur est le français. Les périphériques d'entrée utilisés sont le clavier et la souris.
4. **Caractéristiques du travail :** Le but du travail de l'administrateur est de mettre à jour le système en y incluant les nouveaux utilisateurs. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel.

## Environnement

1. **Localisation du produit :** Le logiciel est utilisé principalement dans les locaux du Club Capra, à l'ETS puisque l'accès à la base de données ne peut se faire que localement.
2. **Position :** L'administrateur est généralement assis pour utiliser l'ordinateur et adopte vraisemblablement cette même position pour utiliser le logiciel.
3. **Matériel :** L'ordinateur de l'administrateur est suffisamment puissant pour faire fonctionner la machine virtuelle java et notre application.
4. **Logiciel et Système d'exploitation :** Le système d'exploitation de l'administrateur est Windows. Il n'y a pas de versions types pour ce système d'exploitation. Une machine virtuelle Java doit être installée.
5. **Structure :**
   - *Travail en groupe :* Le logiciel est utilisé individuellement.
   - *Assistance :* Une aide incorporée au logiciel est disponible pour l'administrateur.
   - *Interruptions :* Ce facteur dépend fortement du contexte dans lequel l'administrateur se trouve. Dans le cas le plus probable, celui-ci n'est pas interrompu.
   - *Communications :* Le réseau de l'ETS permet de véhiculer l'information liée à l'utilisation du logiciel.

## Correcteur

## Caractéristiques

1. **Attributs physiques :** La personne à l'âge légal pour exercer la fonction de correcteur. Il n'y a à priori aucun âge spécifique ni de sexe typique. Rien n'est prévu au sein du logiciel pour la prise en charge d'utilisateurs à capacités réduites. Des incapacités physiques et handicaps modérés sont cependant tolérés.
2. **Attributs mentaux :** Le correcteur est familiarisé aux concepts modernes des interfaces graphiques. Notre application n'est pas adaptée aux handicaps lourds. Du fait de sa qualification et de son intérêt évident pour la revue de code, le correcteur adopte une attitude positive vis-à-vis de la tâche.
3. **Qualifications et connaissances :** La langue maternelle du correcteur est le français. Les périphériques d'entrée utilisés sont le clavier et la souris.
4. **Caractéristiques du travail :** Le but du travail du correcteur est de répertorier dans le système les erreurs qu'il a pu trouver dans les fichiers lors de ses revues de code ainsi que de tenir à jour les ajouts des nouvelles références. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel.

## Environnement

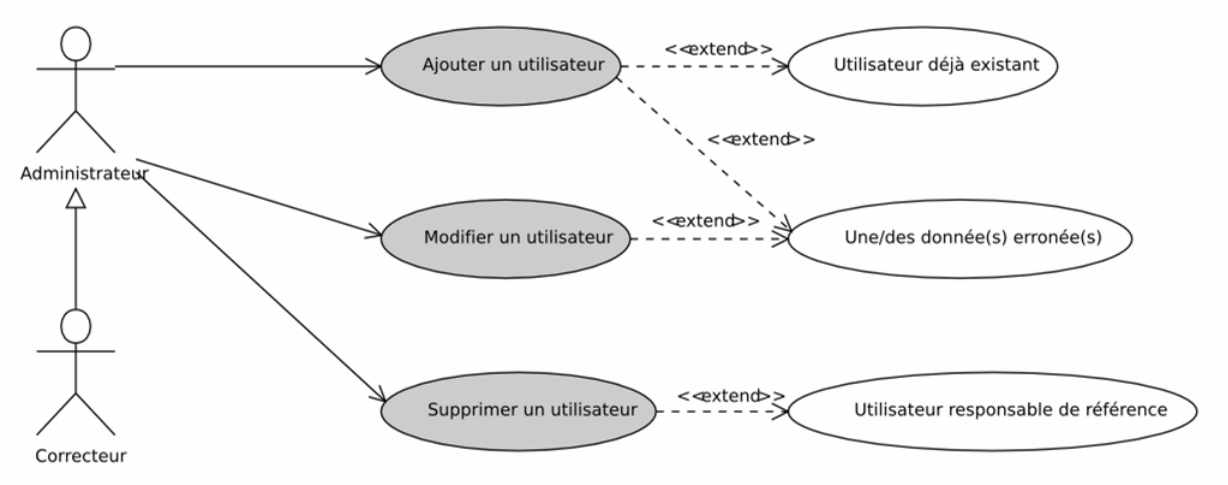
1. **Localisation du produit :** Le logiciel est utilisé principalement dans les locaux du Club Capra, à l'ETS puisque l'accès à la base de données ne peut se faire que localement.
2. **Position :** Le correcteur est généralement assis pour utiliser l'ordinateur et adopte vraisemblablement cette même position pour utiliser le logiciel
3. **Matériel :** L'ordinateur du correcteur est suffisamment puissant pour faire fonctionner la machine virtuelle java et notre application.
4. **Logiciel et Système d'exploitation :** Le système d'exploitation de l'administrateur est Windows. Il n'y a pas de versions types pour ce système d'exploitation. Une machine virtuelle Java doit être installée.
5. **Structure :**
   - *Travail en groupe :* Le logiciel est utilisé individuellement. Si la phase de revue de code se fait en groupe, il est nécessaire de nommer un responsable pour enregistrer les erreurs dans le système.
   - *Assistance :* Une aide incorporée au logiciel est disponible pour le correcteur.
   - *Interruptions :* Ce facteur dépend fortement du contexte dans lequel le correcteur se trouve. Dans le cas le plus probable, celui-ci n'est pas interrompu.
   - *Communications :* Le réseau de l'ETS permet de véhiculer l'information liée à l'utilisation du logiciel.

# Objectifs des utilisateurs

Les Use Cases représentent le comportement affiché par le système sous certaines conditions de manière à satisfaire un objectif de l'un des acteurs. Les Use Case Diagrams montrent quant à eux les acteurs, les limites du système, les relations entre acteurs et le système ainsi que les relations entre les Use Cases eux-mêmes. Ceux-ci permettent, aux travers de scénarii, de capturer, représenter et valider les fonctions d'un domaine, les spécifier et donner un aperçu de la dynamique et des interactions. Du fait de leur simplicité, les Use Cases ont pour avantage notable d'être compréhensibles par tous.

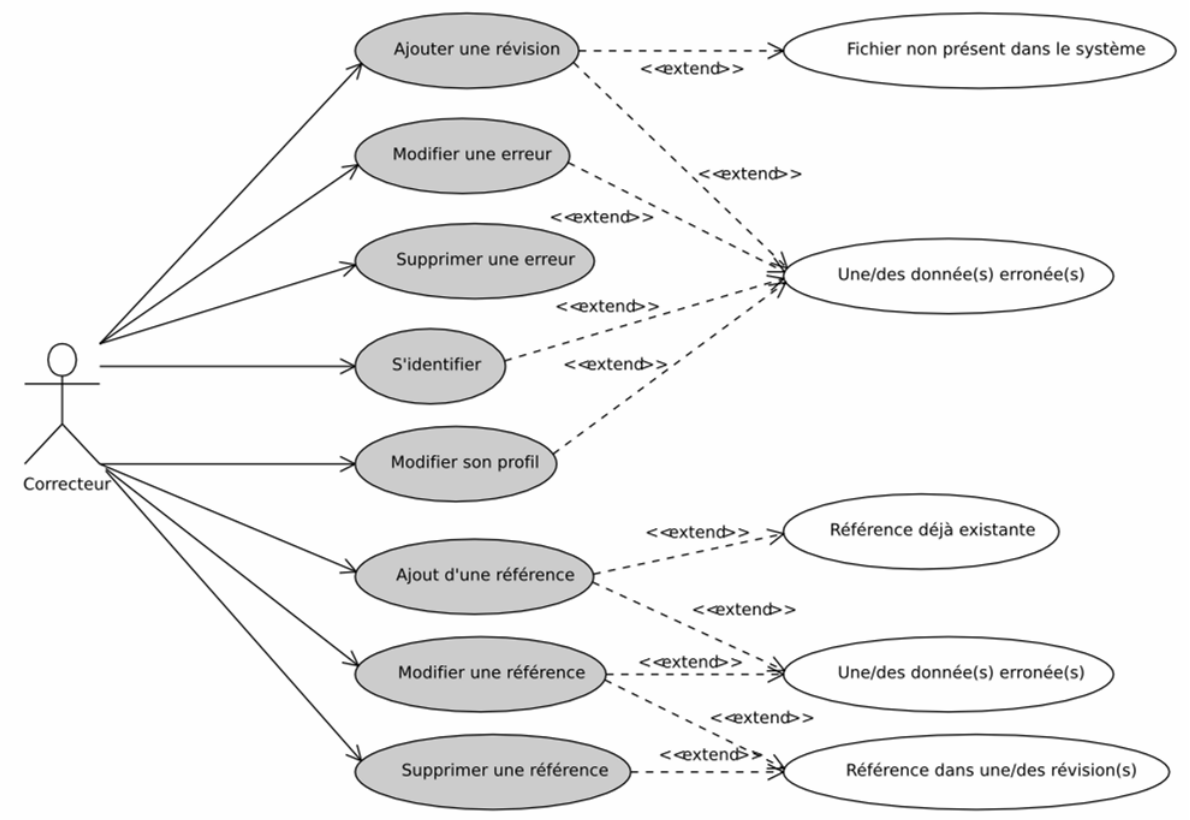
Notons que, afin d'avoir une vision plus précise et plus simple du gestionnaire de revue de code, nous avons divisé ce système en deux parties : le système d'information de l'administrateur (SI administrateur) et le système d'information du correcteur (SI Correcteur).

## SI administrateur



## SI Correcteur

Il est à remarquer que l'administrateur est aussi considéré comme un utilisateur.




Nous allons maintenant décrire les scénarii d'utilisation de l'administrateur.

## Scénarii d'utilisation du côté administrateur

## Ajout d'un utilisateur

### Résumé :

L'administrateur ajoute un nouvel utilisateur dans le système.

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et l'utilisateur est ajouté dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur entre les caractéristiques du nouvel utilisateur et les envoie au système. | 2. Le système reçoit les données de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. L'administrateur est averti que le nouvel utilisateur a bien été ajouté au système avec un login déterminé. |

## Extension : Une/des donnée(s) erronée(s)

### Résumé :

L'administrateur ajoute un nouvel utilisateur dans le système avec une/des donnée(s) incorrecte(s).

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel et une/des donnée(s) est/sont incorrecte(s).

### Postcondition :

Le système est opérationnel et aucun utilisateur n'a été ajouté dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur entre les caractéristiques du nouvel utilisateur et les envoie au système avec une/des donnée(s) incorrecte(s). | |
| | 2. Le système reçoit les données de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. L'administrateur est averti que certaines données sont incorrectes.<br>5. Le système propose à l'administrateur de modifier les caractéristiques de l'utilisateur qui doit être rajouté. |
| 6. Retour au point 2 du cas normal. | |

## Extension : Utilisateur déjà existant

### Résumé :

L'administrateur ajoute un nouvel utilisateur dans le système qui est déjà enregistré dans le système.

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel et l'utilisateur est déjà enregistré dans le système.

### Postcondition :

Le système est opérationnel et le nouvel utilisateur n'est pas enregistré dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur entre les caractéristiques du nouvel utilisateur et les envoie au système. | 2. Le système reçoit les données de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. L'administrateur est averti que le nouvel utilisateur est déjà inscrit dans le système.<br>5. Le système propose à l'administrateur de modifier les caractéristiques de l'utilisateur qui doit être rajouté. |
| 6. Retour au point 2 du cas normal. | |

## Modifier un utilisateur

### Résumé :

L'administrateur modifie certaines caractéristiques d'un utilisateur.

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel et l'utilisateur est enregistré dans le système.

### Postcondition :

Le système est opérationnel et certaines caractéristiques de l'utilisateur ont été changées.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur entre les nouvelles caractéristiques de l'utilisateur et les envoie au système. | 2. Le système reçoit les données de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. L'administrateur est averti que les changements ont été effectués. |

## Extension : Une/des donnée(s) erronée(s)

### Résumé :

L'administrateur modifie de façon erronée certaines caractéristiques d'un utilisateur.

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel et l'utilisateur est enregistré dans le système.

### Postcondition :

Le système est opérationnel et les caractéristiques de l'utilisateur reste inchangée(s).

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur entre de façon erronée les nouvelles caractéristiques de l'utilisateur et les envoie au système. | 2. Le système reçoit les données de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. L'administrateur est averti que les données sont erronée(s).<br>5. Le système propose à l'administrateur de modifier les caractéristiques de l'utilisateur qui doivent être modifiées. |
| 6. Retour au point 2 du cas normal. | |

## Supprimer un utilisateur

### Résumé :

L'administrateur supprime un des utilisateurs du système.

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel et l'utilisateur est enregistré dans le système.

### Postcondition :

Le système est opérationnel et l'utilisateur n'est plus enregistré dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur sélectionne l'utilisateur à supprimer et envoie les données au système. | 2. Le système reçoit les données de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. L'administrateur est averti que l'utilisateur est bien supprimé. |

## Extension : Utilisateur responsable de références

### Résumé :

L'administrateur supprime un des utilisateurs du système mais celui-ci est responsable de certaines références.

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel et l'utilisateur est enregistré dans le système et est responsable de certaines références.

### Postcondition :

Le système est opérationnel et l'utilisateur n'est plus enregistré dans le système. Ses références sont attribuées à un autre utilisateur.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur sélectionne l'utilisateur à supprimer et envoie les données au système. | |
| | 2. Le système reçoit les données de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. L'administrateur est averti que l'utilisateur est responsable de certaines références.<br>5. Le système propose à l'administrateur de sélectionner un nouveau responsable. |
| 6. L'administrateur choisit un nouveau responsable et envoie les données au système. | |
| | 7. Le système reçoit les données de l'administrateur.<br>8. Le système traite et vérifie les données.<br>9. L'administrateur est averti que l'utilisateur est supprimé et que les références sont bien allouées à un autre responsable. |

## Lister les utilisateurs

### Résumé :

Lister les utilisateurs permet à l'administrateur de connaître l'ensemble des utilisateurs, et leurs caractéristiques, enregistrés dans le système.

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et l'administrateur peut prendre connaissance des caractéristiques des utilisateurs enregistrés dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur demande la liste de tous les utilisateurs enregistrés dans le système. | 2. Le système reçoit la demande de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. La liste des utilisateurs est renvoyée à l'administrateur. |

## Exporter la liste des utilisateurs

### Résumé :

Exporter la liste des utilisateurs permet à l'administrateur d'avoir, indépendamment du programme, l'ensemble des utilisateurs, et leurs caractéristiques, enregistrés dans le système.

### Acteur :

L'administrateur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et l'administrateur possède l'ensemble des utilisateurs, et leurs caractéristiques, indépendamment du programme.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur demande d'exporter la liste de tous les utilisateurs enregistrés dans le système. | 2. Le système reçoit la demande de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. La liste des utilisateurs a été exportée et est à disposition de l'administrateur. |

Nous allons maintenant décrire les scénarii d'utilisation du correcteur.

## S'identifier

### Résumé :

Le correcteur doit s'identifier afin d'accéder au programme de gestion de revue de code.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et le correcteur est identifié.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur lance le système.<br>2. Le correcteur entre son login et son mot de passe. | <br><br><br>3. Le système reçoit les données du correcteur.<br>4. Le système traite et vérifie les données. |

## Extension : Login et/ou mot de passe erroné(s)

### Résumé :

Le correcteur s'identifie avec un login et/ou un mot de passe erroné(s).

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et le correcteur entre un login et/ou un mot de passe erroné(s).

### Postcondition :

Le système est opérationnel.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur lance le système.<br>2. Le correcteur entre son login et son mot de passe de façon erronée. | |
| | 3. Le système reçoit les données du correcteur.<br>4. Le système traite et vérifie les données.<br>1. Le système signale au correcteur qu'il a rentré un login et/ou un mot de passe erroné(s).<br>2. Le système redemande au correcteur de s'identifier. |
| 3. Retour au point 2 du cas normal. | |

## Extension : Correcteur non administrateur

### Résumé :

Le correcteur s'identifie comme administrateur sans que son compte le lui permette.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et le correcteur s'identifie comme administrateur.

### Postcondition :

Le système est opérationnel et le correcteur est averti que son compte ne lui permet pas de se connecter comme administrateur.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur lance le système.<br>2. Le correcteur entre son login et son mot de passe et tente de se loger comme administrateur. | |
| | 3. Le système reçoit les données du correcteur.<br>4. Le système traite et vérifie les données.<br>1. Le système signale au correcteur que son compte ne possède pas les droits d'administrateur.<br>2. Le système redemande au correcteur de s'identifier. |
| 3. Retour au point 2 du cas normal. | |

## Récupération du mot de passe

### Résumé :

Le correcteur ne se souvient plus de son mot de passe et tente donc de le récupérer

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et le correcteur ne connait plus son mot de passe.

### Postcondition :

Le système est opérationnel et le mot de passe est renvoyé au correcteur.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur lance le système.<br>2. Ne se souvenant plus de son mot de passe, le correcteur demande son mot de passe au système. | 3. Le système reçoit les données du correcteur.<br>4. Le système traite et vérifie les données.<br>5. Le mot de passe du correcteur lui est renvoyé. |

Pour les cas d'utilisation suivants, nous considérerons que le correcteur est identifié au système.

## Modifier son profil

### Résumé :

Le correcteur modifie son profil d'utilisateur.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et le correcteur a devant lui les caractéristiques de son profil actuel.

### Postcondition :

Le système est opérationnel et le profil du correcteur a été modifié.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre les données afin de modifier son profil. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que son profil a été modifié avec succès. |

## Extension : Une/des donnée(s) erronée(s).

### Résumé :

Le correcteur modifie son profil d'utilisateur et entre une/des donnée(s) incorrecte(s).

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et une/des donnée(s) entrée(s) par le correcteur est/sont incorrecte(s).

### Postcondition :

Le système est opérationnel et le profil du correcteur reste inchangé.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre une/des données incorrecte(s) afin de modifier son profil. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que son profil est inchangé car une/des données est/sont incorrecte(s).<br>5. Le système demande au correcteur de rentrer des données correctes. |

## Ajouter une révision

### Résumé :

Le correcteur insère dans le système la phase de revue de code qu'il vient de réaliser

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et le système contient les fichiers inclus dans la phase de revue de code.

### Postcondition :

Le système est opérationnel et les données ont été rentrées dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre les données relatives à sa phase de revue de code et les envoie au système. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que ses données ont bien été enregistrées. |

## Extension : une/des données sont erronée(s)

### Résumé :

Le correcteur insère dans le système la phase de revue de code qu'il vient de réaliser avec une/des donnée(s) erronée(s).

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel, le système contient les fichiers inclus dans la phase de revue de code et une/des donnée(s) est/sont erronée(s).

### Postcondition :

Le système est opérationnel et les données restent inchangée(s).

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre des données erronée(s) relative(s) à sa phase de revue de code et les envoie au système. | |
| | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que ses données sont erronée(s).<br>5. Le système demande au correcteur de corriger les donnée(s) erronée(s). |
| 6. Retour au point 2 du cas normal. | |

## Extension : Référence(s) de la révision non présente(s) dans le système

### Résumé :

Le correcteur insère dans le système la phase de revue de code qu'il vient de réaliser mais une/des références non inclue(s) dans le système.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et certaines références ne sont pas enregistrée(s) dans le système.

### Postcondition :

Le système est opérationnel et les références sont enregistrées dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre des données et les envoie au système. | |
| | 2. Le système reçoit les données du correcteur. |
| | 3. Le système traite et vérifie les données. |
| | 4. Le système s'aperçoit que certaines références ne sont pas enregistrées et en prévient le correcteur. |
| 5. Le correcteur demande au système d'enregistrer ses références. | |
| | 6. Le système reçoit les données du correcteur. |
| | 7. Le système traite et vérifie les données. |
| 6. Retour au point 2 du cas normal. | |

## Modifier une erreur

### Résumé :

Le correcteur modifie une erreur répertoriée dans le système.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et l'erreur est modifiée.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre les données modifiées de l'erreur et les envoie au système. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que les données ont bien été modifiées. |

## Extension : une/des donnée(s) erronée(s)

### Résumé :

Le correcteur modifie une erreur répertoriée dans le système et une/des donnée(s) est/sont erronée(s).

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et certaines données sont erronée(s).

### Postcondition :

Le système est opérationnel et le correcteur est averti que une/des donnée(s) est/sont erronée(s).

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre les données modifiées de l'erreur de façon erronée et les envoie au système. | |
| | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que certaines données sont erronée(s) et demande au correcteur de les corriger. |
| 3. Retour au point 2 du cas normal. | |

## Suppression d'une erreur

### Résumé :

Le correcteur supprime une erreur enregistrée dans le système.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et l'erreur a été supprimée.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur sélectionne l'erreur à supprimer et l'envoie au système. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que l'erreur a bien été supprimée. |

## Lister les erreurs

### Résumé :

Lister les erreurs permet au correcteur de connaître l'ensemble des erreurs, et leurs caractéristiques, enregistrées dans le système.

### Acteur :

Le correcteur

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et le correcteur peut prendre connaissance des caractéristiques des erreurs enregistrées dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur demande la liste des erreurs enregistrées dans le système. | 2. Le système reçoit la demande de l'administrateur.<br>3. Le système traite et vérifie les données.<br>4. La liste des erreurs est renvoyée au correcteur. |

## Exporter la liste des erreurs

### Résumé :

Exporter la lister des erreurs permet au correcteur d'avoir, indépendamment du programme, l'ensemble des erreurs, et leurs caractéristiques, enregistrées dans le système.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et le correcteur possède l'ensemble des erreurs, et leurs caractéristiques, indépendamment du programme.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur demande d'exporter la liste de toutes les erreurs enregistrées dans le système. | 2. Le système reçoit la demande du correcteur.<br>3. Le système traite et vérifie les données.<br>4. La liste des erreurs a été exportée et est à disposition du correcteur. |

## Ajout d'une référence

### Résumé :

Le correcteur ajoute une nouvelle référence dans le système.

### Acteur :

Le correcteur

### Précondition :

Le système est opérationnel et la référence n'est pas encore présente dans le système.

### Postcondition :

Le système est opérationnel et la référence est ajouté dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre les caractéristiques de la nouvelle référence et les envoie au système. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que la nouvelle référence a bien été ajoutée au système. |

## Extension : Une/des donnée(s) erronée(s)

### Résumé :

Le correcteur ajoute une nouvelle référence dans le système avec une/des donnée(s) incorrecte(s).

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et une/des donnée(s) est/sont incorrecte(s).

### Postcondition :

Le système est opérationnel et aucune référence n'a été ajoutée dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'administrateur entre les caractéristiques de la nouvelle référence et les envoie au système avec une/des donnée(s) incorrecte(s). | |
| | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que certaines données sont incorrectes.<br>5. Le système propose au correcteur de modifier les caractéristiques de la référence qui doit être rajoutées. |
| 6. Retour au point 2 du cas normal. | |

## Extension : Référence déjà existante

### Résumé :

Le correcteur ajoute une nouvelle référence qui est déjà enregistrée dans le système.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et la référence est déjà enregistrée dans le système.

### Postcondition :

Le système est opérationnel et la nouvelle référence n'est pas enregistrée dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre les caractéristiques de la nouvelle référence et les envoie au système. | |
| | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que la nouvelle référence est déjà inscrite dans le système.<br>5. Le système propose au correcteur de modifier les caractéristiques de la référence qui doit être rajoutée. |
| 6. Retour au point 2 du cas normal. | |

## Modifier une référence

### Résumé :

Le correcteur modifie certaines caractéristiques d'une référence.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et la référence est enregistrée dans le système.

### Postcondition :

Le système est opérationnel et les caractéristiques de la référence ont été modifiées.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre les nouvelles caractéristiques de la référence et les envoie au système. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que les changements ont été effectués. |

## Extension : Une/des donnée(s) erronée(s)

### Résumé :

Le correcteur modifie de façon erronée certaines caractéristiques d'une référence.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel et le correcteur est enregistré dans le système.

### Postcondition :

Le système est opérationnel et les caractéristiques de la référence restent inchangée(s).

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur entre de façon erronée les nouvelles caractéristiques de la référence et les envoie au système. | |
| | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que les données sont erronée(s).<br>5. Le système propose au correcteur de modifier les caractéristiques de la référence qui doivent être modifiées. |
| 6. Retour au point 2 du cas normal. | |

## Extension : Référence inclue dans une/des révisions

### Résumé :

Le correcteur modifie une des références du système mais celle-ci est inclue dans une/des révision(s) enregistrée(s) dans le système.

### Acteur :

Le correcteur

### Précondition :

Le système est opérationnel et la référence est enregistrée dans le système ainsi que dans une/des révision(s) enregistrée(s) dans le système.

### Postcondition :

Le système est opérationnel et la référence a été modifiée. Les révisions incluant cette référence ont été mises à jour.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur sélectionne la référence à modifier et envoie les données au système. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le système modifie la référence et met à jour les révisions contenant celle-ci.<br>6. Le correcteur est averti que la référence est modifiée et que les révisions sont bien mise à jour suite à cette modification. |

## Supprimer une référence

### Résumé :

Le correcteur supprime une des références enregistrées du système.

### Acteur :

Le correcteur

### Précondition :

Le système est opérationnel et la référence est enregistrée dans le système.

### Postcondition :

Le système est opérationnel et la référence n'est plus enregistrée dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur sélectionne la référence à supprimer et envoie les données au système. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le correcteur est averti que la référence est bien supprimée. |

## Extension : Référence inclue dans une/des révisions

### Résumé :

Le correcteur supprime une des références du système mais celle-ci est inclue dans une/des révision(s) enregistrée(s) dans le système.

### Acteur :

Le correcteur

### Précondition :

Le système est opérationnel et la référence est enregistrée dans le système ainsi que dans une/des révision(s) enregistrée(s) dans le système.

### Postcondition :

Le système est opérationnel et la référence n'est plus enregistrée dans le système. Les révisions incluant cette référence ont été mises à jour.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur sélectionne la référence à supprimer et envoie les données au système. | 2. Le système reçoit les données du correcteur.<br>3. Le système traite et vérifie les données.<br>4. Le système supprime la référence et met à jour les révisions incluant celle-ci.<br>5. Le correcteur est averti que la référence est supprimée et que les révisions sont bien mises à jour suite à cette suppression. |

## Lister les références

### Résumé :

Lister les références permet au correcteur de connaître l'ensemble des références, et leurs caractéristiques, enregistrées dans le système.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et le correcteur peut prendre connaissance des caractéristiques des références enregistrées dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur demande la liste de toutes les références enregistrées dans le système. | 2. Le système reçoit la demande du correcteur.<br>3. Le système traite et vérifie les données.<br>4. La liste des références est renvoyée au correcteur. |

## Exporter la liste des références

### Résumé :

Exporter la lister des références permet au correcteur d'avoir, indépendamment du programme, l'ensemble des références, et leurs caractéristiques, enregistrés dans le système.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et le correcteur possède l'ensemble des références, et leurs caractéristiques, indépendamment du programme.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. Le correcteur demande d'exporter la liste de toutes les références enregistrées dans le système. | 2. Le système reçoit la demande du correcteur.<br>3. Le système traite et vérifie les données.<br>4. La liste des références a été exportée et est à disposition du correcteur. |

## Consulter l'aide

### Résumé :

Le correcteur demande de l'aide sur la manière dont fonctionne le programme.

### Acteur :

Le correcteur.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel et le correcteur reçoit l'aide demandée.

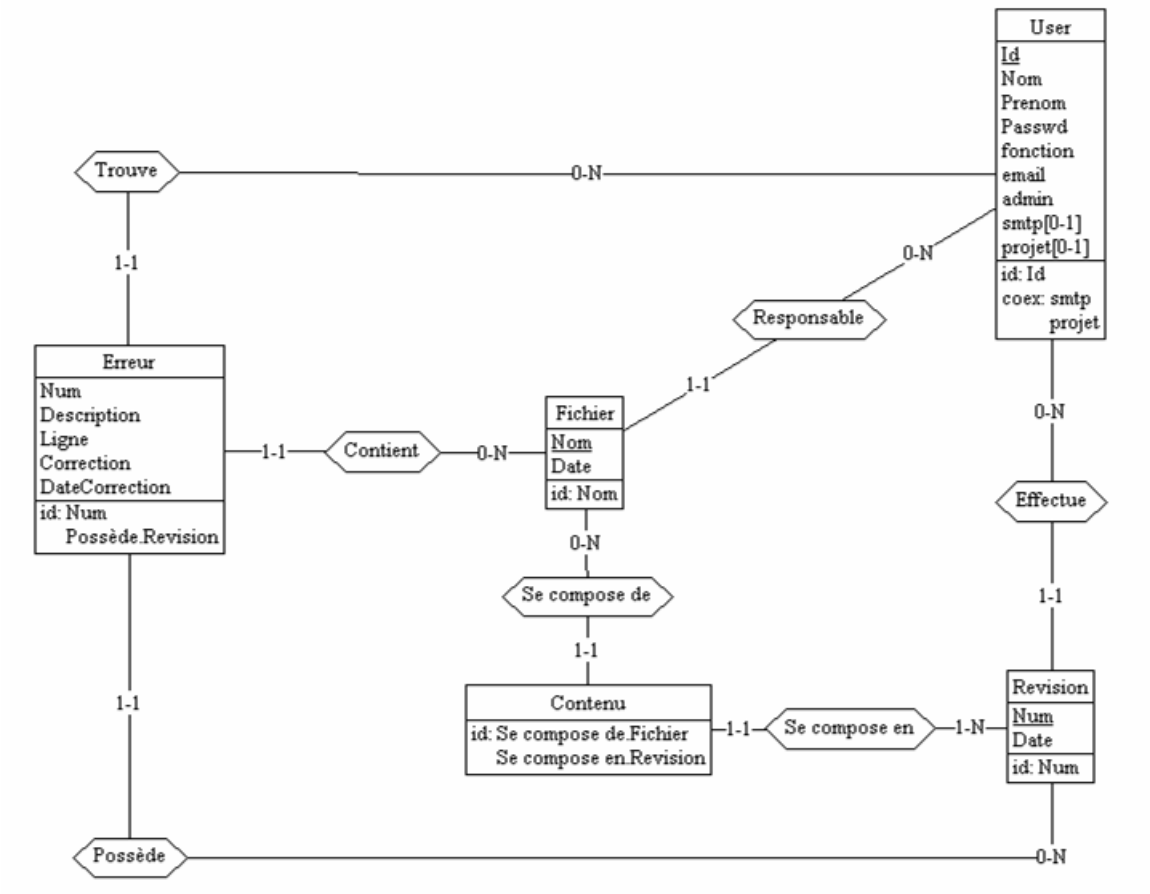
### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. le correcteur demande de l'aide sur la manière dont fonctionne le programme. | 2. Le système reçoit la demande de l'administrateur.<br>3. Le système traite la demande.<br>4. L'aide est renvoyée au correcteur. |

# Schéma de la statique des données

## Schéma de la statique

Le modèle Entité-Relation-Attribut propose des concepts (principalement les entités, les associations et les attributs) permettant de décrire un ensemble de données relatives à un domaine défini.



## Description des tables

## Table User

La table User contient les caractéristiques des utilisateurs enregistrés dans le système et se compose des attributs suivants :

- *Id :* Il représente l'identifiant de l'utilisateur. C'est une valeur numérique de 6 chiffres qui est unique.
- *Nom :* Il représente le nom de l'utilisateur.
- *Prénom :* Il représente le prénom de l'utilisateur.
- *Passwd :* Il représente le mot de passe nécessaire à l'utilisateur afin de s'identifier au démarrage.
- *Fonction :* Il représente la fonction de l'utilisateur au sein du Club Capra.
- *Email :* Il représente l'adresse e-mail de l'utilisateur nécessaire à l'envoi des mails le concernant.
- *Admin :* Il représente le statut de l'utilisateur dans le programme : un administrateur (valeur = 1) ou un simple utilisateur (valeur = 2).
- *smtp :* Il représente le serveur SMTP de l'utilisateur. Il ne possède pas de valeur tant que l'utilisateur ne s'est pas connecté au système pour la première fois.
- *Projet :* Il représente le path du répertoire du projet de l'utilisateur. Il ne possède pas de valeur tant que l'utilisateur ne s'est pas connecté au système pour la première fois.

## Table Fichier

La table Fichier contient les caractéristiques des références enregistrées dans le système et se compose des attributs suivants :

- *Nom :* Il représente l'identifiant d'une référence. Il est constitué de l'ensemble des packages parents de la référence et du nom de celle-ci.
- *Date :* Il représente la date d'insertion de la référence dans le système.

## Table Revision

La table Revision contient les caractéristiques des révisions enregistrées dans le système et se compose des attributs suivants :

- *Num :* Il représente l'identifiant de la révision. C'est une valeur numérique qui est unique.
- *Date :* Il représente la date à laquelle la révision est effectuée.

## Table Erreur

La table Erreur contient les caractéristiques des erreurs enregistrées dans le système et se compose des attributs suivants :

- *Num :* Il représente l'identifiant de l'erreur. C'est une valeur numérique qui est unique.
- *Description :* Il représente la description de l'erreur détectée.
- *Ligne :* Il représente le numéro de ligne du fichier où l'erreur a été détectée.
- *Correction :* Il représente le statut de l'erreur, à savoir si elle a déjà été corrigée ou non.
- *DateCorrection :* Il représente la date pour laquelle l'erreur doit être corrigée.

# Fonctionnalités du système

Dans cette dernière partie de l'analyse des besoins, nous avons classés l'ensemble des fonctionnalités souhaitées par le Club Capra en deux parties[7] :

## Fonctionnalités primaires

- **S'identifier :** Permet de rentrer dans le système, soit comme administrateur, soit comme correcteur. Seul l'administrateur a le droit de rentrer dans les deux parties du système.
- **Ajouter un utilisateur :** Permet à l'administrateur de rentrer un nouvel utilisateur dans le système.
- **Modifier un utilisateur :** Permet à l'administrateur de modifier un utilisateur enregistré dans le système.
- **Supprimer un utilisateur :** Permet à l'administrateur de supprimer un utilisateur enregistré dans le système.
- **Ajouter une référence :** Permet au correcteur d'ajouter une nouvelle référence (un fichier) dans le système.
- **Modifier une référence :** Permet au correcteur de modifier une référence enregistrée dans le système.
- **Supprimer une référence :** Permet au correcteur de supprimer une référence dans le système.
- **Ajouter une révision :** Permet au correcteur d'ajouter une nouvelle phase de révision de code.
- **Modifier une erreur :** Permet au correcteur de modifier une des erreurs d'une des phases de révision.
- **Supprimer une erreur :** Permet au correcteur de supprimer une erreur qui n'a plus d'intérêt.
- **Modifier mon profil :** Permet à l'administrateur et au correcteur de modifier leur profil.
- **Signaler les corrections :** Permet de signaler au correcteur lors de sa connexion que certaines des références dont il est responsable contiennent des erreurs qui doivent être impérativement corrigées.

[7] Un administrateur est aussi un correcteur

## Fonctionnalité secondaires

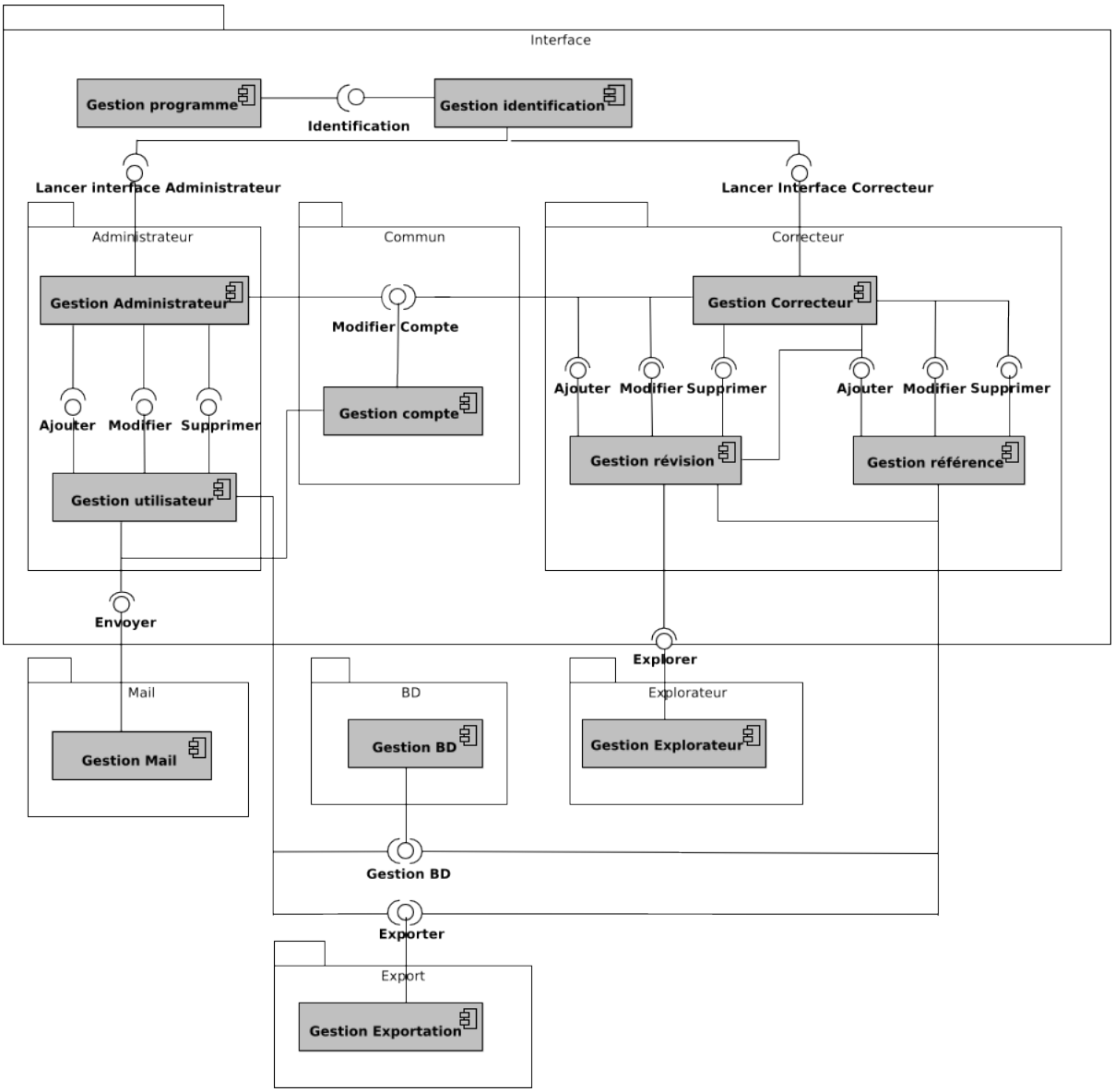
- **Consulter l'aide :** Permet à l'administrateur et au correcteur de consulter le manuel d'aide.
- **Récupérer le mot de passe :** Permet à l'administrateur et au correcteur de récupérer leur mot de passe via l'envoi d'un e-mail.
- **Lister les utilisateurs :** Permet à l'administrateur de connaître l'ensemble des utilisateurs présents dans le système ainsi que leur caractéristiques.
- **Exporter les utilisateurs :** Permet à l'administrateur d'exporter au format pdf la liste des utilisateurs.
- **Lister les références :** Permet au correcteur de connaître l'ensemble des références présentes dans le système ainsi que leur caractéristiques.
- **Exporter les références :** Permet au correcteur d'exporter au format pdf la liste des références.
- **Lister les erreurs :** Permet au correcteur de connaître l'ensemble des erreurs présentes dans le système ainsi que leur caractéristiques.
- **Exporter les erreurs :** Permet au correcteur d'exporter au format pdf la liste des erreurs.
- **Passer en mode utilisateur :** Permet à l'administrateur de passer en mode correcteur sans devoir s'identifier à nouveau.
- **Passer en mode administrateur :** Permet au correcteur, s'il possède des droits d'administrateur, de passer en mode administrateur sans devoir s'identifier à nouveau.
- **Configuration :** Permet à l'administrateur et au correcteur de changer son serveur SMTP ainsi que le path de son projet.

# Rapport C :

*Capra - Cahier d'implémentation*

# Diagramme de composants

Le diagramme de composants décrit l'organisation du système du point de vue des modules de code. Ce diagramme permet de mettre en évidence les dépendances entre les composants (qui utilise quoi) et permet ainsi de mieux organiser les modules. L'encapsulation (insérer un composant dans un autre, cacher des détails d'implémentation,...) permet de réduire la complexité et d'élever le niveau d'abstraction. Le système développé pour Club Capra peut donc être représenté par le diagramme de composants suivant :



*Gestion Programme*

Ce premier composant a pour fonction de lancer le gestionnaire de revue de code. Une fois sa tâche terminée, il passe la main au gestionnaire d'identification.

*Gestion Identification*

Ce gestionnaire a pour fonction la gestion d'identification des différents types d'utilisateurs, à savoir l'administrateur et le correcteur. Une fois l'administrateur identifié, il passe la main au gestionnaire de l'administrateur et dans le cas du correcteur, au gestionnaire du correcteur.

*Gestion Administrateur*

Ce gestionnaire gère l'interface de l'administrateur. Il passera la main au gestionnaire de compte si l'administrateur invoque une fonctionnalité relative la gestion de son compte ou au gestionnaire d'utilisateurs si l'utilisateur invoque une fonctionnalité relative à la gestion des utilisateurs.

*Gestion Correcteur*

Ce gestionnaire gère l'interface du correcteur. Il passera la main au gestionnaire de compte si le correcteur invoque une fonctionnalité relative à la gestion de son compte, au gestionnaire de révisions si le correcteur invoque une fonctionnalité relative à la gestion des révisions ou au gestionnaire de références si le correcteur invoque une fonctionnalité relative à la gestion des références.

*Gestion Utilisateur*

Ce gestionnaire gère les fonctionnalités relatives à la gestion des utilisateurs, à savoir : ajouter, modifier, supprimer, lister et exporter des utilisateurs. Pour leur bon fonctionnement, ce gestionnaire peut faire appel au gestionnaire

- de la base de donnée pour manipuler des données existantes ou en insérer des nouvelles,
- d'exportation afin d'exporter au format pdf certaines caractéristiques des utilisateurs enregistrés dans le système,
- de gestion des mails pour envoyer des informations aux utilisateurs si leur compte a été créé, modifié ou supprimé.

*Gestion Compte*

Ce gestionnaire gère les fonctionnalités relatives au compte utilisateur, à savoir modifier son compte, passer en mode administrateur/correcteur, se déconnecter ou quitter le programme. Pour les mêmes raisons que le composant "Gestion Utilisateur", il peut faire appel aux gestionnaires de la base de données et de gestion des mails.

*Gestion Révision*

Ce gestionnaire gère les fonctionnalités relatives aux phases de révision de code, à savoir : ajouter, modifier, supprimer, lister ou exporter des révisions. Pour leur bon fonctionnement, ce gestionnaire peut faire appel au gestionnaire

- de la base de données pour manipuler des données existantes ou en insérer de nouvelles,
- de l'explorateur pour parcourir le projet associé à la phase de révision de code et y sélectionner les références à y inclure,
- des références pour enregistrer si nécessaire les références non encore présentes dans la base de données,
- d'exportation afin d'exporter au format pdf certaines caractéristiques des phases de révision de code enregistrées dans le système.

*Gestion Référence*

Ce gestionnaire gère les fonctionnalités relatives aux références, à savoir : ajouter, modifier, supprimer, lister ou exporter des références. Pour leur bon fonctionnement, ce gestionnaire peut faire appel au gestionnaire de la base de donnée pour manipuler des données existantes ou en insérer des nouvelles et d'exportation afin d'exporter au format pdf certaines caractéristiques des références enregistrées dans le système.

*Gestion Mail*

Ce gestionnaire s'occupe de l'envoi par e-mail des informations qui lui sont procurées par le gestionnaire appelant. Le mail est envoyé à l'utilisateur spécifié par ce même gestionnaire appelant.

*Gestion BD*

Ce gestionnaire s'occupe de la base de données. Il regroupe les fonctions d'accès à la base de données et de gestion de son contenu (ajouter des données, récupérer des données, modifier des données,...).

*Gestion Export*

Ce gestionnaire s'occupe d'exporter les données demandées par le gestionnaire appelant dans un rapport au format pdf selon certaines caractéristiques fournies par ce même gestionnaire appelant.

*Gestion Explorateur*

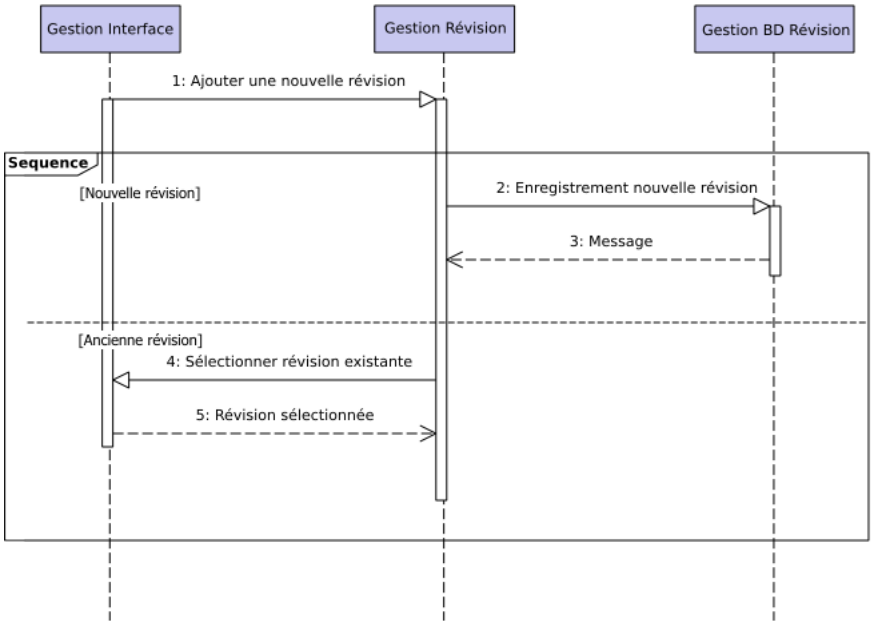
Ce gestionnaire offre au gestionnaire appelant un explorateur de fichier selon un chemin déterminé sous forme d'un arbre comprenant des répertoires et des fichiers. Les noeuds représentent les packages du projet et les feuilles, les fichiers.

# Diagrammes de séquence

## Enregistrement d'une phase de révision

L'enregistrement d'une nouvelle phase de révision de code étant la fonctionnalité principale du système mais aussi la plus difficile à implémenter, nous l'avons représentée à l'aide de diagrammes de séquence. Ce processus d'enregistrement peut donc être divisé en 3 phases ; l'enregistrement ou la sélection de la phase de révision de code, l'enregistrement des références si nécessaire et l'enregistrement des erreurs détectées.

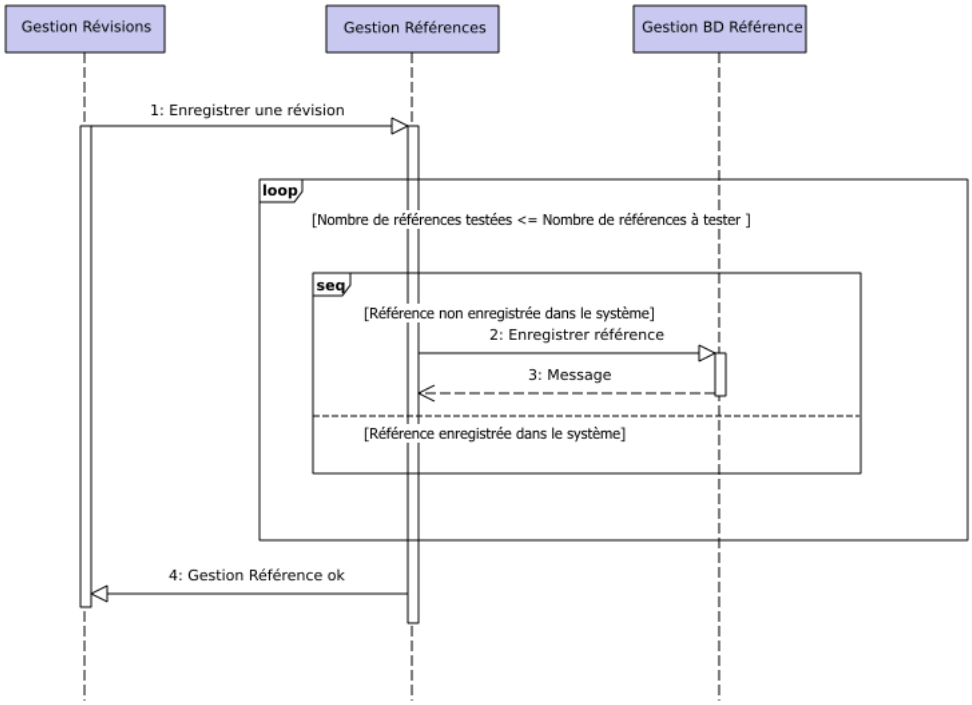
### Gestion de la révision




La première étape consiste à indiquer si les erreurs détectées lors de la phase de révision sont à ajouter à une ancienne phase ou à une nouvelle. Dans le premier cas, le gestionnaire d'interface reprend la main afin que l'utilisateur choisisse une phase de révision existante. Ensuite, il va renvoyer ce choix sous lequel les erreurs devront être répertoriées. Dans le deuxième cas, une nouvelle révision est ajoutée au système[8] .

[8] Afin de rendre le schéma plus compréhensible, certains choix ont été entrepris :
- Gestion Interface est une généralisation de Gestion Administrateur et Gestion Correcteur du diagramme de composants et
- Gestion BD Révision est une spécification de Gestion BD du diagramme de composants.

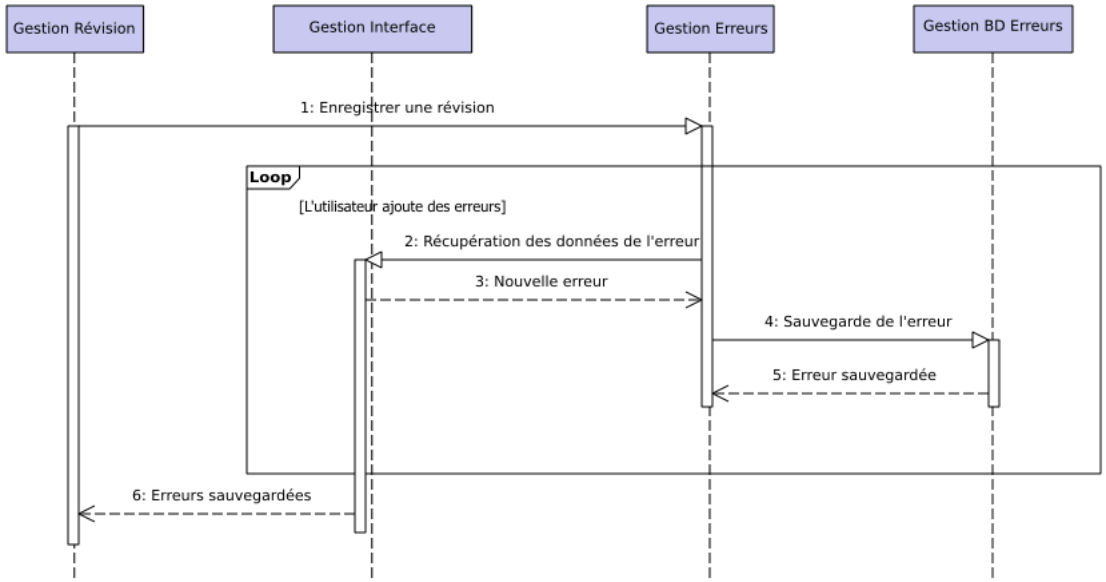
### Gestion des références




Une fois la gestion de la révision effectuée, on passe à la gestion des références. En effet, il est possible que certaines références n'aient pas encore été enregistrées dans le système alors qu'elles sont inclues dans la phase de révision de code en cours d'enregistrement. C'est pourquoi, le gestionnaire de références va tester pour chaque référence si celle-ci est déjà enregistrée dans le système ou non. Si elle ne l'est pas encore, elle y sera automatique enregistrée[9].

[9]Gestion BD Référence est une spécification de Gestion BD du diagramme de composants.

### Gestion des erreurs



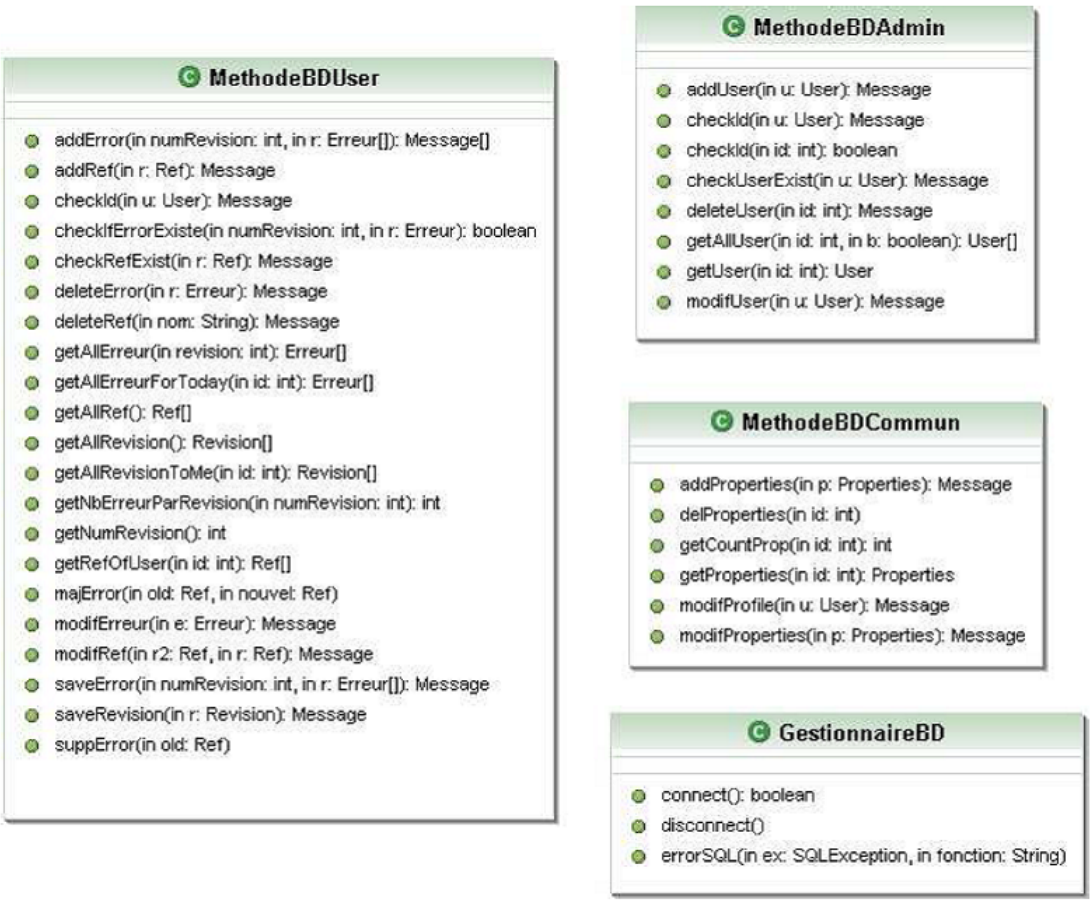

La dernière étape consiste à enregistrer les erreurs détectées lors de la phase de révision de code. C'est pourquoi, tant que l'utilisateur ajoute des erreurs via son interface, le gestionnaire d'erreurs va récupérer les éléments spécifiés par l'utilisateur et ajouter la nouvelle erreur dans le système[10].

[10] Afin de rendre le schéma plus compréhensible, certains choix ont été entrepris :
- Gestion Interface est une généralisation de Gestion Administrateur et Gestion Correcteur du diagramme de composants,
- Gestion BD Erreurs est une spécification de Gestion BD du diagramme de composants et
- Gestion Erreurs est une spécialisation de Gestion Révision du diagramme de composants.

# Diagramme de classes

## Package BD

Ce package comprend l'ensemble des fonctions utilisées pour ajouter, modifier, supprimer, récupérer des éléments dans la base de données.

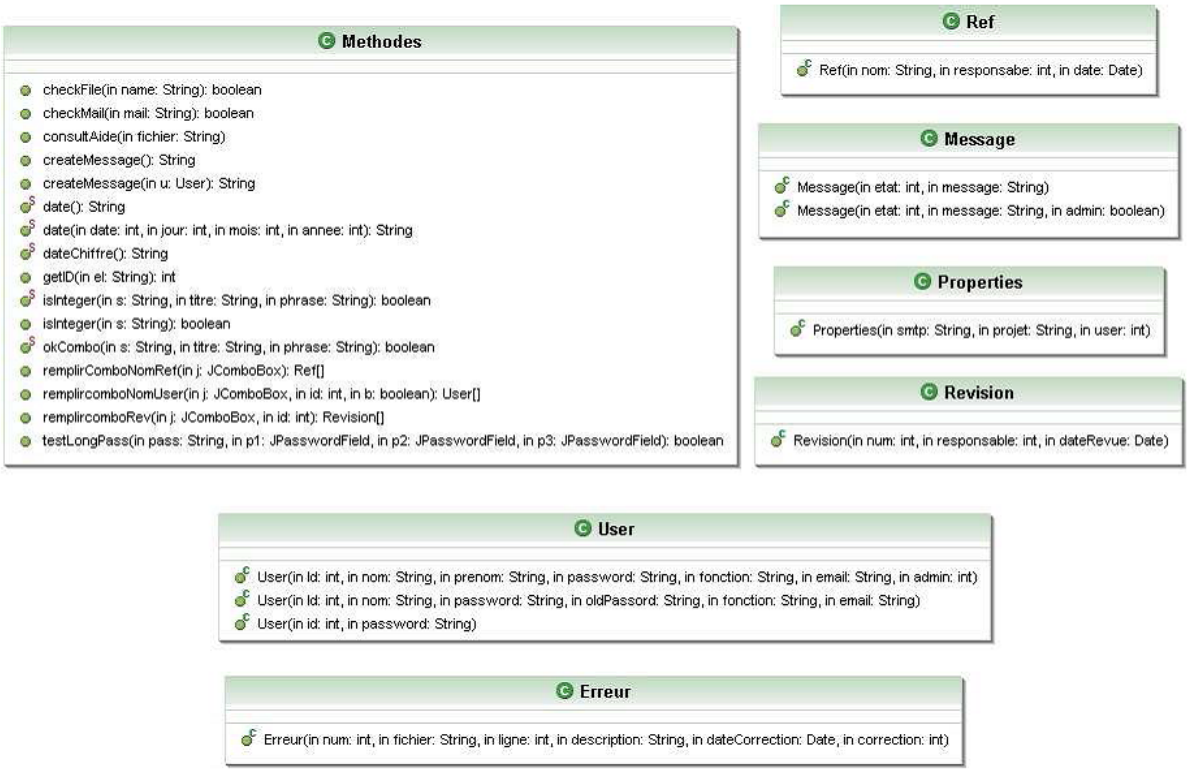


## Package Calendrier

Ce package comprend l'ensemble des fonctions utilisées pour afficher un calendrier.

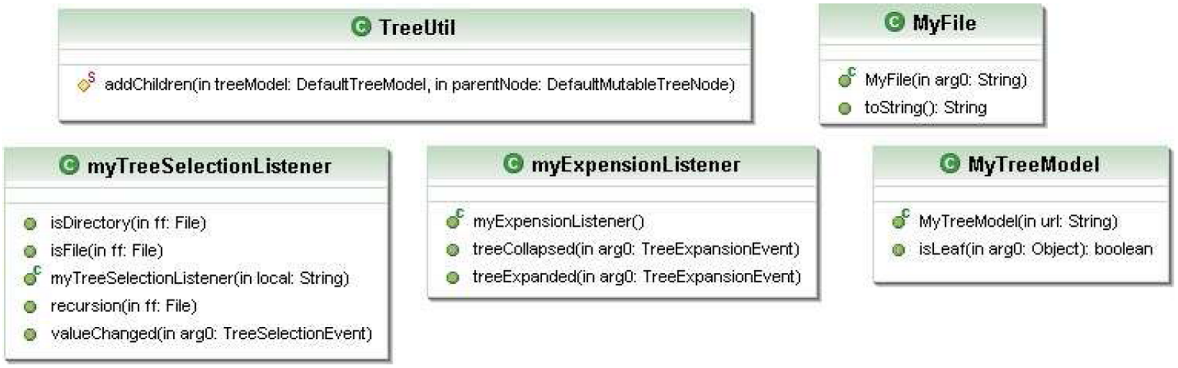

## Package Commun

Ce package comprend l'ensemble des fonctions utilisées pour instancier les objets et traiter les données.



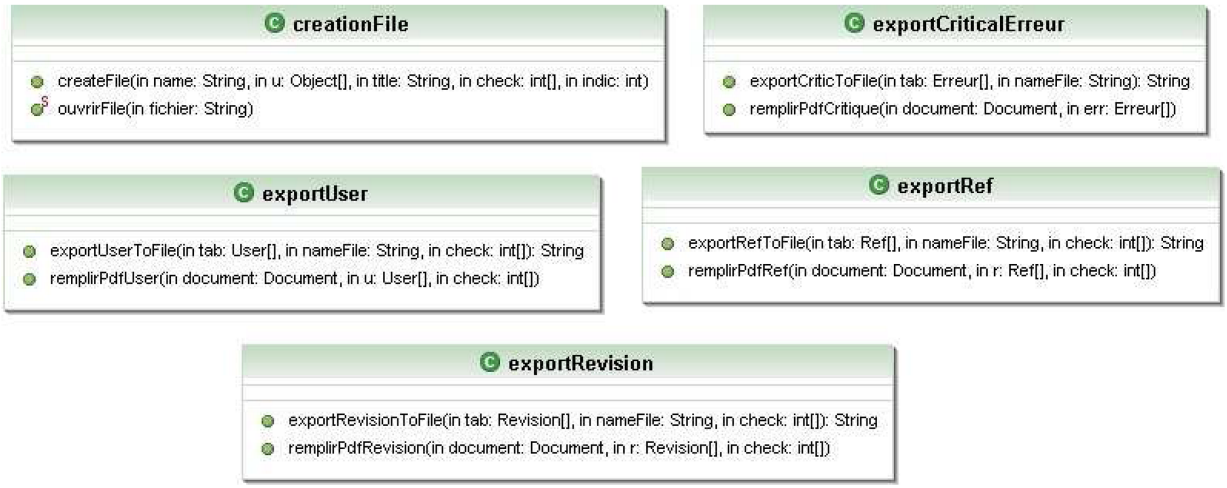
## Package Explorateur

Ce package comprend l'ensemble des fonctions utilisées pour afficher un explorateur de fichiers dans les interfaces.




## Package Export

Ce package comprend l'ensemble des fonctions utilisées pour exporter les documents au format pdf.

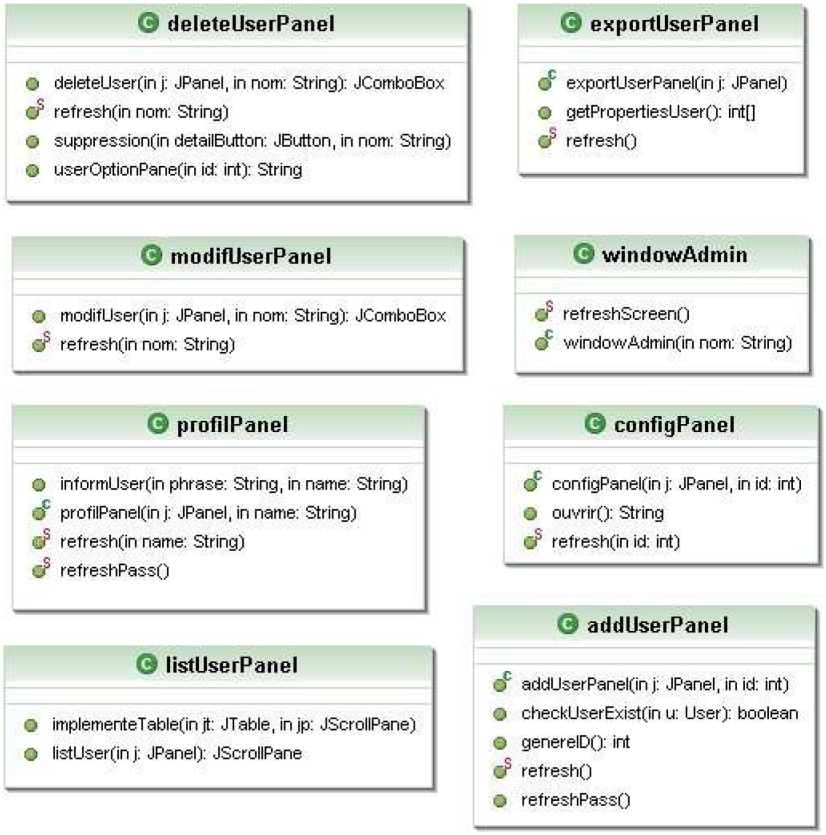


## Package Interface

Ce package comprend l'ensemble des fonctions utilisées pour lancer le gestionnaire de revue de code.

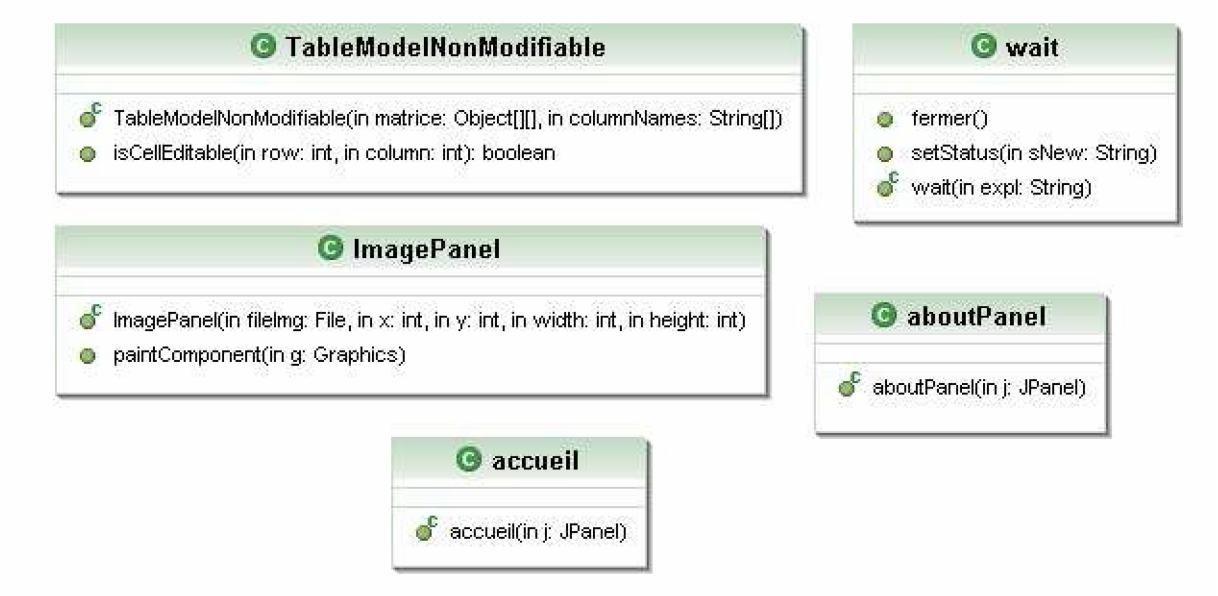

## Package Interface.Admin

Ce package comprend l'ensemble des fonctions utilisées pour afficher l'interface de l'administrateur.



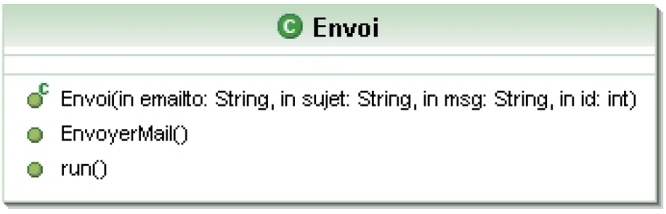
## Package Interface.Commun

Ce package comprend l'ensemble des fonctions utilisées pour afficher les parties d'interface communes à l'administrateur et l'utilisateur (le correcteur).



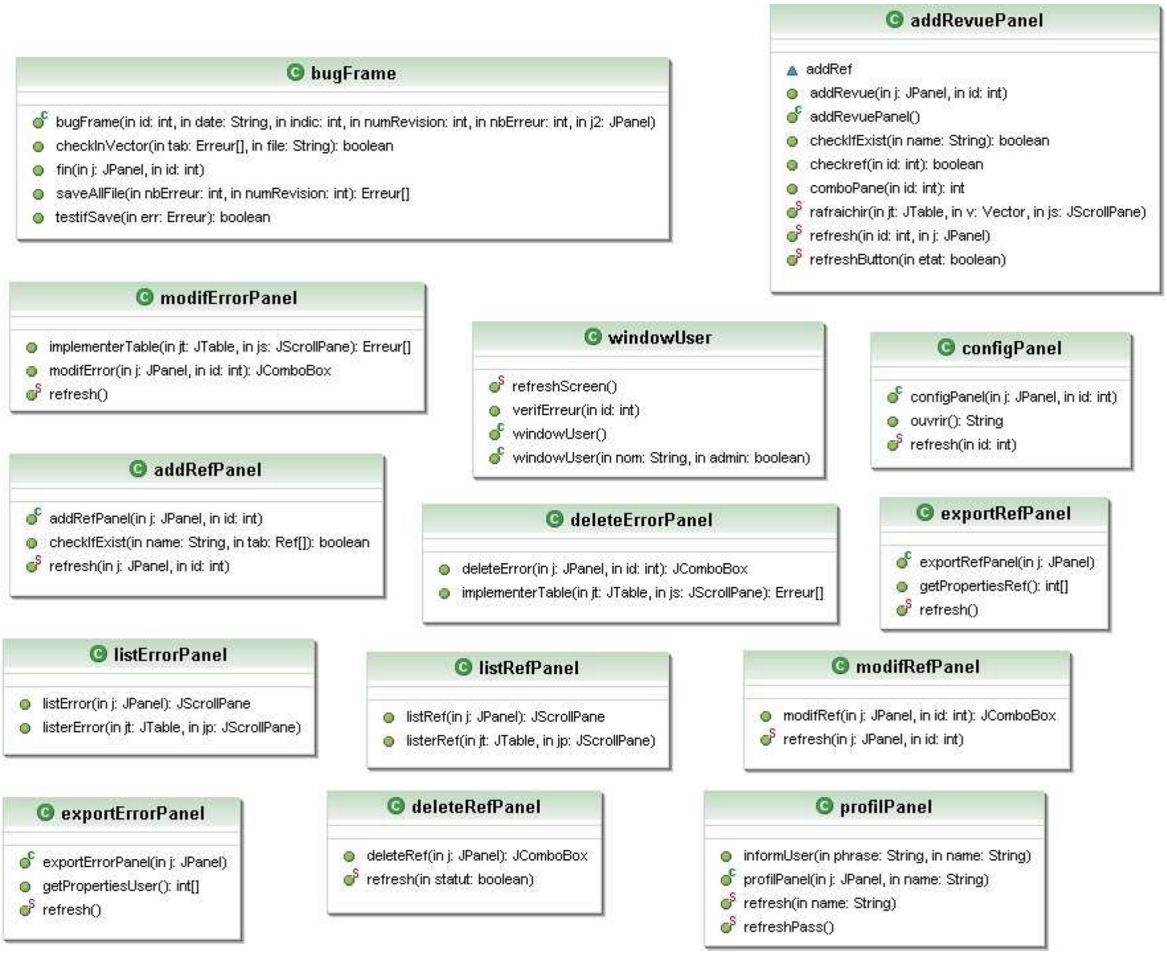

## Package Mail

Ce package comprend l'ensemble des fonctions utilisées pour l'envoi d'e-mails.



## Package Interface.User

Ce package comprend l'ensemble des fonctions utilisées pour afficher l'interface de l'utilisateur (le correcteur).



# Validation et vérification

Cette dernière partie est constituée d'un ensemble de tests réalisés sur le gestionnaire de revue de code destiné à Capra.

## Interface login

## S'identifier comme administrateur

### Cas normal

- *Description :* L'administrateur entre son login et son mot de passe et tente de se connecter au système comme administrateur.
- *Résultat attendu :* L'administrateur est rentré dans le système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Login et/ou mot de passe invalide(s)

- *Description :* L'administrateur entre un mauvais login et/ou mot de passe et tente de se connecter au système comme administrateur.
- *Résultat attendu :* L'administrateur est averti que son login et/ou mot de passe est/sont incorrect(s).
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## S'identifier comme utilisateur

### Cas normal

- *Description :* Le correcteur entre son login et son mot de passe et tente de se connecter au système comme correcteur.
- *Résultat attendu :* Le correcteur est rentré dans le système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Login et/ou mot de passe invalide(s)

- *Description :* Le correcteur entre un mauvais login et/ou mot de passe et tente de se connecter au système comme correcteur.
- *Résultat attendu :* Le correcteur est averti que son login et/ou mot de passe est/sont incorrect(s).
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Compte sans droits d'administrateur

- *Description :* Le correcteur entre son login et son mot de passe et tente de se connecter au système comme administrateur, son compte n'ayant pas les droits d'administrateur.
- *Résultat attendu :* Le correcteur est averti qu'il ne possède pas les droits d'administrateur sur son compte.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Récupération de mot de passe

### Cas normal

- *Description :* L'utilisateur[11] demande qu'on lui envoie son mot de passe.
- *Résultat attendu :* Le mot de passe est renvoyé sur sa boîte e-mail.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

[11] Utilisateur = Administrateur ou Correcteur

# Interface Administrateur

# Modifier mon profil

### Cas normal

- *Description :* L'administrateur modifie son profil
- *Résultat attendu :* Le profil de l'administrateur est modifié.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* L'administrateur entre mal son ancien mot de passe et/ou son nouveau mot de passe et/ou une adresse e-mail invalide et/ou une confirmation du nouveau mot de passe différente du nouveau mot de passe.
- *Résultat attendu :* L'administrateur est averti qu'une/des donnée(s) est/sont incorrecte(s).
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

# Ajouter un utilisateur

### Cas normal

- *Description :* L'administrateur ajoute un nouvel utilisateur dans le système.
- *Résultat attendu :* L'utilisateur est ajouté dans le système et celui-ci reçoit un mail contenant les caractéristiques de son compte.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* L'administrateur ajoute un nouvel utilisateur dans le système mais avec une/des donnée(s) incorrecte(s).
- *Résultat attendu :* L'administrateur est averti qu'une/des données est/sont incorrecte(s).
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Utilisateur déjà existant

- *Description :* L'administrateur ajoute un nouvel utilisateur dans le système mais celui-ci existe déjà.
- *Résultat attendu :* L'administrateur est averti que l'utilisateur est déjà enregistré dans le système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Modifier un utilisateur

### Cas normal

- *Description :* L'administrateur modifie les caractéristiques d'un des utilisateurs, à savoir sa fonction ou son statut.
- *Résultat attendu :* Le profil de l'utilisateur est mis à jour et les nouvelles caractéristiques entrées sont envoyées à l'utilisateur concerné par mail.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* L'administrateur modifie les caractéristiques d'un des utilisateurs, à savoir sa fonction ou son statut, avec une/des donnée(s) incorrecte(s).
- *Résultat attendu :* L'administrateur est averti qu'une/des données est/sont incorrecte(s).
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Supprimer un utilisateur

### Cas normal

- *Description :* L'administrateur supprime le compte d'un utilisateur.
- *Résultat attendu :* Le compte utilisateur est supprimé. Un mail est envoyé à l'utilisateur afin de l'en avertir.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Utilisateur responsable de références

- *Description :* L'administrateur supprime le compte d'un utilisateur mais celui-ci est responsable de références.
- *Résultat attendu :* L'administrateur est invité à choisir un nouveau responsable pour ses références. Les références sont ensuite mises à jour et l'utilisateur est averti par mail que son compte a été supprimé.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Lister les utilisateurs

### Cas normal

- *Description :* L'administrateur demande la liste de tout les utilisateurs enregistré dans le système.
- *Résultat attendu :* L'administrateur peut consulter la liste de tout les utilisateurs enregistrés dans le système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Exporter les utilisateurs

### Cas normal

- *Description :* L'administrateur demande d'exporter la liste des utilisateurs suivant les caractéristiques et le nom du fichier qu'il a déterminé.
- *Résultat attendu :* Le fichier pdf est créé et l'administrateur peut l'ouvrir via l'interface.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* L'administrateur demande d'exporter la liste des utilisateurs mais aucune caractéristique n'est sélectionnée et/ou aucun nom de fichier n'est entré.
- *Résultat attendu :* L'administrateur est averti qu'aucune caractéristique n'est sélectionnée et/ou qu'aucun nom de fichier n'a été donné.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Consulter l'aide

### Cas normal

- *Description :* L'administrateur consulte l'aide fournit avec le système.
- *Résultat attendu :* L'administrateur voit le manuel d'aide s'ouvrir devant lui.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Interface Correcteur

## Modifier mon profil

### Cas normal

- *Description :* Le correcteur modifie son profil.
- *Résultat attendu :* Le profil du correcteur est modifié.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* Le correcteur entre mal son ancien mot de passe et/ou son nouveau mot de passe et/ou une adresse e-mail invalide et/ou une confirmation du nouveau mot de passe différente du nouveau mot de passe.
- *Résultat attendu :* Le correcteur est averti qu'une/des donnée(s) est/sont incorrecte(s).
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Ajouter une révision

### Cas normal - Nouvelle révision

- *Description :* Le correcteur ajoute une nouvelle révision au système incluant plusieurs fichiers et erreurs
- *Résultat attendu :* La révision et les erreurs la constituant sont bien enregistrés dans le système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Cas normal - Révision existante

- *Description :* Le correcteur ajoute des erreurs à une révision qu'il a déjà commencée.
- *Résultat attendu :* Les erreurs sont ajoutées à la révision en cours.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* Le correcteur ajoute des erreurs à une révision qu'il a déjà commencée ou à une nouvelle avec une/des erreur(s) telles qu'une mauvaise date, une mauvaise ligne ou une mauvaise description.
- *Résultat attendu :* Le correcteur est averti qu'il y a une/des donnée(s) incorrecte(s).
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Duplication d'erreurs

- *Description :* Le correcteur ajoute des erreurs à une nouvelle ou ancienne révision mais enregistre plusieurs fois la même erreur.
- *Résultat attendu :* Le correcteur est averti qu'il a rentré deux fois la même erreur et continue ensuite d'enregistrer les erreurs suivantes.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Erreur(s) déjà existante(s) - Révision existante

- *Description :* Le correcteur ajoute des erreurs à une révision avec une/des erreur(s) déjà existantes.
- *Résultat attendu :* Le correcteur est averti qu'il y a une/des erreur(s) déjà enregistrée(s) et que le reste des données entrées a été sauvegardé.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Modifier une révision

### Cas normal

- *Description :* Le correcteur modifie une erreur d'une révision qu'il sélectionne préalablement dans une liste.
- *Résultat attendu :* L'erreur est modifiée.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* Le correcteur modifie une erreur mais se trompe dans la date limite de correction.
- *Résultat attendu :* Le correcteur est averti que la date de correction est mauvaise.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### L'erreur a déjà été corrigée

- *Description :* Le correcteur veut modifier une erreur qui a déjà été corrigée.
- *Résultat attendu :* Le correcteur est averti que l'erreur ne peut plus être modifiée car elle a déjà été corrigée.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Supprimer une erreur d'une révision

### Cas normal

- *Description :* Le correcteur supprime une erreur contenue dans une des révisions qu'il a effectuées.
- *Résultat attendu :* L'erreur est supprimée de la révision.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Lister les révisions

### Cas normal

- *Description :* L'administrateur demande la liste des révisions et des erreurs, ainsi que leurs caractéristiques, enregistrées dans le système.
- *Résultat attendu :* L'administrateur reçoit la liste des erreurs et révisions sur son interface.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Exporter les révisions

### Cas normal

- *Description :* Le correcteur demande d'exporter la liste des révisions et erreurs suivant les caractéristiques et le nom du fichier qu'il a déterminé.
- *Résultat attendu :* Le fichier pdf est créé et le correcteur peut l'ouvrir via l'interface.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* Le correcteur demande d'exporter la liste des révisions et erreurs mais aucune caractéristique n'est sélectionnée et/ou aucun nom de fichier n'est entré.
- *Résultat attendu :* L'administrateur est averti qu'aucune caractéristique n'est sélectionnée et/ou qu'aucun nom de fichier n'a été donné.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Ajouter une référence

### Cas normal

- *Description :* Le correcteur ajoute une nouvelle référence dans le système
- *Résultat attendu :* La référence est ajoutée dans le système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* Le correcteur ajoute une nouvelle référence dans le système mais avec une/des donnée(s) incorrecte(s).
- *Résultat attendu :* Le correcteur est averti qu'une/des données est/sont incorrecte(s).
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Référence déjà existante

- *Description :* Le correcteur ajoute un nouvelle référence dans le système mais celle-ci existe déjà.
- *Résultat attendu :* Le correcteur est averti que la référence est déjà enregistrée dans le système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Modifier une référence

### Cas normal

- *Description :* Le correcteur modifie les caractéristiques d'une référence enregistrée dans le système, à savoir le nom du fichier et/ou sons emplacement.
- *Résultat attendu :* La référence est bien modifiée.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* Le correcteur modifie une référence enregistrée dans le système mais avec une/des donnée(s) incorrecte(s).
- *Résultat attendu :* Le correcteur est averti qu'une/des données est/sont incorrecte(s). La référence n'est pas modifiée.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Référence déjà existante

- *Description :* Le correcteur modifie une référence enregistrée dans le système mais celle-ci existe déjà.
- *Résultat attendu :* Le correcteur est averti que la référence est déjà enregistrée dans le système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Référence inclue dans une/des révision(s)

- *Description :* Le correcteur modifie les caractéristiques d'une référence enregistrée dans le système, à savoir le nom du fichier et/ou sons emplacement. De plus, cette référence est inclue dans une/des révision(s).
- *Résultat attendu :* La référence est bien modifiée et les révisions sont mises à jour.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Supprimer une référence

### Cas normal

- *Description :* L'administrateur supprime une référence enregistrée dans le système.
- *Résultat attendu :* La référence est supprimée du système.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Référence inclue dans une/des révision(s)

- *Description :* Le correcteur supprime une référence enregistrée dans le système qui est référencée dans une révision.
- *Résultat attendu :* La référence est bien supprimée et les erreurs la référençant aussi.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Lister les références

### Cas normal

- *Description :* Le correcteur demande la liste des références enregistrées dans le système.
- *Résultat attendu :* Le correcteur reçoit la liste de références ainsi que les caractéristiques de celles-ci.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Exporter les références

### Cas normal

- *Description :* Le correcteur demande d'exporter la liste des références suivant les caractéristiques et le nom du fichier qu'il a déterminé.
- *Résultat attendu :* Le fichier pdf est créé et le correcteur peut l'ouvrir via l'interface.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

### Une/des donnée(s) incorrecte(s)

- *Description :* Le correcteur demande d'exporter la liste des références mais aucune caractéristique n'est sélectionnée et/ou aucun nom de fichier n'est entré.
- *Résultat attendu :* L'administrateur est averti qu'aucune caractéristique n'est sélectionnée et/ou qu'aucun nom de fichier n'a été donné.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

## Consulter l'aide

### Cas normal

- *Description :* Le correcteur consulte l'aide fournit avec le système.
- *Résultat attendu :* Le correcteur voit le manuel d'aide s'ouvrir devant lui.
- *Résultat final :* Il correspond au résultat attendu. Le test est donc réussi.

# Rapport D :

*Horizon Vert - Plan d'action*

# Avant-propos

Ce document consiste en une synthèse détaillée du plan d'action élaboré pour l'entreprise Horizon Vert Centre-du-Québec. Ce rapport ce base principalement sur le rapport de la micro-évaluation réalisé sur Horizon Vert le 25 juin 2004. Suite à cette première analyse, nous avons fait ressortir 4 problèmes clés, présentés de la priorité la plus haute à la plus basse :

1. **Gestion des exigences :** L'entreprise ne formalise pas ses exigences puisqu'elle ne travaille pas avec un cahier des charges et un processus de modification d'exigence est déclenché par un simple coup de téléphone.
2. **Gestion de la documentation :** L'entreprise ne possède pas de gabarit pour structurer les documents d'analyses et les livrables.
3. **Gestion de projet :** Pas de gestion formelle de projet ni d'estimation formelle pour les projets d'envergure.
4. **Méthodologie de développement :** Horizon Vert ne possède pas de cycle de vie formalisé ni de méthodologie de développement de logiciel.

Ce que nous pouvons réaliser pour Horizon Vert doit dépendre du court terme puisqu'il nous est seulement possible de déployer une application sur 2 mois. Il semble donc logique que le problème de méthodologie de développement soit écarté puisqu'il se réfère à du moyen terme. De plus, l'objectif de notre intervention est d'implémenter un outil amenant le plus de valeur ajoutée possible dans l'entreprise. C'est pourquoi, le problème de gestion de projet peut être repoussé à plus tard puisqu'il est déjà implémenté partiellement dans l'entreprise et ne se réfère donc pas à des problèmes généraux.

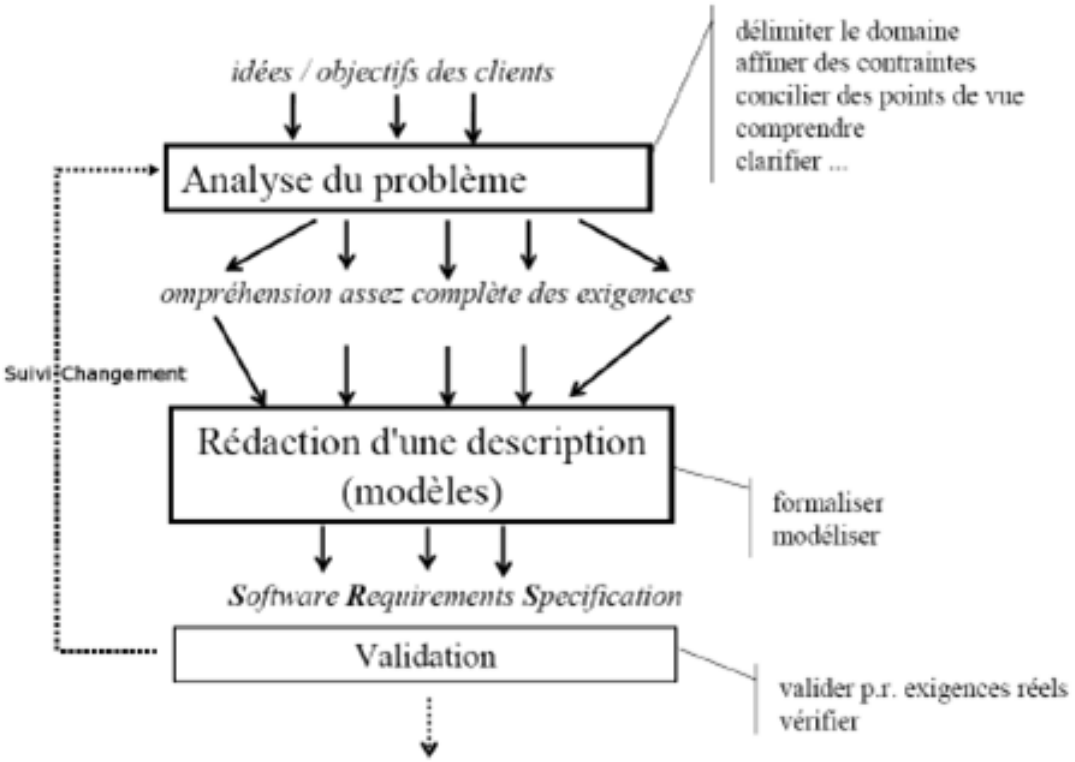
Nous proposons donc pour Horizon Vert de prendre en charge le problème de **gestion des exigences** qui nous semble être le plus critique à l'heure actuelle. Si le temps nous le permet, nous pourrons alors nous pencher sur la gestion de sa **documentation** afin de la rendre plus formelle.

La suite du document reprend les différents problèmes décelés chez Horizon Vert ainsi qu'une possibilité de solution à ces problèmes. Ils sont présentés du plus prioritaire au moins prioritaire.

# Gestion des exigences

## Son importance

Une bonne gestion des exigences implique une bonne formalisation de celles-ci. Cette formalisation doit contenir tous les éléments nécessaires afin d'apporter une bonne description et une bonne compréhension des besoins du client. De plus, la gestion de la documentation, qui consiste le deuxième problème d'Horizon Vert se verrait améliorée. Un point important à retenir est que si les exigences ne sont pas bien spécifiées ou pas bien comprises, les erreurs peuvent alors coûter cher une fois arrivé à la fin du projet.




## Notre solution

Afin de rendre plus formelle la description des exigences, il est possible d'utiliser un ensemble de programmes qui mettent à la disposition de l'utilisateur une "carte" pour décrire les exigences. Par exemple, la description des exigences fonctionnelles et non fonctionnelles du système peut se réaliser via le template Volere qui offre une carte de description des exigences. On

pourrait alors travailler avec un programme qui permettrait à l'utilisateur de remplir ses fiches et qui donnerait un rapport reprenant l'ensemble des fiches entrées. Le choix du programme reste quant à lui à déterminer suivant les éléments de description des exigences souhaitées par Horizon Vert.

<table>
<tr><td>Exigence # : identifiant unique</td><td>Type d'exigence : référence au modèle</td><td>Evénements : Liste d'événements qui nécessitent cette exigence</td></tr>
<tr><td colspan="3">Description : objectif de l'exigence en une phrase</td></tr>
<tr><td colspan="3">Justification : raison pour laquelle cette exigence est présente</td></tr>
<tr><td colspan="3">Origine : qui a émis cette exigence ?</td></tr>
<tr><td colspan="3">Critère de satisfaction : une mesure de l'exigence qui permet de tester si la solution proposée remplit l'exigence initiale.</td></tr>
<tr><td colspan="2">Points positifs : degré de contentement si le produit final satisfait cette exigence. De 1 (pas intéressé) à 5 (très content).</td><td>Points négatifs : degré de mécontentement si le produit final ne satisfait pas cette exigence. De 1 (quasi indifférent) à 5 (très mécontent)</td></tr>
<tr><td colspan="2">Exigences dépendantes : liste d'exigences dont l'implémentation dépend de l'implémentation de celle-ci.</td><td>Exigences conflictuelles : exigences qui ne peuvent pas être implémentées si celle-ci l'est.</td></tr>
<tr><td colspan="3">Documents relatifs : référence à des documents qui illustrent et expliquent cette exigence.</td></tr>
<tr><td colspan="3">Historique : création, modifications, destruction apportées à l'exigence.</td></tr>
</table>

Numéroter les exigences sert à tracer les exigences au cours du processus de développement.

- Le numéro de l'exigence est un numéro unique, choisi de manière séquentielle.
- Le type d'exigence est le numéro de la rubrique du cahier des charges dans laquelle sera classé ce type d'exigence. Exemple : numéro 9 pour les exigences fonctionnelles.
- Les degrés de contentement et de mécontentement du maître d'ouvrage sont à demander au maître d'ouvrage.

# Gestion de la documentation

L'objectif de documenter son projet est de promouvoir l'essor du projet, c'est-à-dire :
- S'assurer de la qualité du concept
- Garder le cap du projet
- Garantir un suivi efficace
- Soutenir une évaluation sérieuse
- Obtenir une interprétation commune
- Permettre une meilleure communication au sein de l'équipe de projet

Bien sûr, la documentation n'apporte pas que des avantages au projet lui-même car elle permet aussi :
- D'étendre le projet
- De reproduire le projet
- De partager les leçons apprises
- De partager de nouveaux outils

Le processus de documentation se déroule durant toutes les phases du projet : il se retrouve lors de l'analyse des besoins avec un cahier des charges, lors de la phase de conception avec des diagrammes ou lors de la phase de test avec des rapports d'évaluation. C'est pourquoi, une bonne description des exigences est une excellente base pour la documentation du projet.

Pour améliorer la gestion de la documentation d'Horizon Vert, il est difficile de définir directement une solution sans avoir préalablement rencontré les dirigeants. C'est en effet grâce à la première interview que nous pourrons évaluer comment Horizon Vert documente ses activités, quelles sont les activités qui doivent être documentées,...

# Gestion de projet

On appelle projet l'ensemble des actions à entreprendre afin de répondre à un besoin défini dans des délais fixés. Ainsi un projet étant une action temporaire avec un début et une fin, mobilisant des ressources identifiées (humaines et matérielles) durant sa réalisation, celui-ci possède également un coût et fait donc l'objet d'une budgétisation de moyens et d'un bilan indépendant de celui de l'entreprise.


La difficulté dans la conduite du projet réside en grande partie dans la multiplicité des acteurs qu'il mobilise. La clé du succès d'un projet réside dans les mots ‘ *"Diviser pour régner"* c'est-à-dire que le projet doit être divisé en sous-ensemble dont la complexité est plus facilement maîtrisable. Une bonne gestion de projet consiste donc en une bonne estimation des délais, des coûts, des exigences,... pour chaque sous-ensemble. Ces estimations peuvent être réalisées via quelques indicateurs tels que :

- Utilisation des ressources (en %)
- Tâches réalisées/tâches planifiées
- Jalons
- Date de fin initiale du projet et des livrables
- Date de fin finale du projet et des livrables
- Avancement en délai (%)
- Nombre de tâches terminées par rapport au nombre de tâches prévues
- Nombre de changements
- Nombre de risques réalisés

Horizon Vert Centre-du-Québec ne possède pas de mécanisme d'estimation formelle ni de gestion formelle de projet. La résolution de ce problème semble cependant être moins primordiale puisqu'elle ne concernerait que les projets d'envergures. Il nous est alors difficile de connaître la véritable ampleur de ce problème puisque nous ne savons pas quel est le pourcentage des projets qu'Horizon Vert qualifie d'envergure.

# Méthodologie de développement

L'entreprise Horizon Vert Centre-du-Québec ne possède pas de méthodologie de développement pour les différents projets qu'elle réalise. Nous pouvons donc supposer que l'entreprise développe ses projets en se basant sur son expérience et les membres de son équipe de projet. Ce problème se réfère à du moyen terme et il ne nous est donc pas possible de le résoudre puisque nous n'avons que deux mois pour implémenter nos solutions.

Il faut cependant remarquer que ce problème est important puisque ce n'est qu'avec une bonne méthodologie de développement qu'on arrive à développer des projets correctement, dans les délais demandés par le client et avec une bonne gestion des ressources.

Si la résolution de ce problème est cruciale pour Horizon Vert, il nous est possible de recommander l'entreprise à un autre professeur de l'ETS qui pourrait alors, dans le cadre d'un de ses cours, prendre en charge l'implémentation d'une méthodologie de développement.

# Bibliographie


[1] **Micro-évaluation**, *Rapport de micro-évaluation des pratiques logiciels - Horizon Vert*, Mohammed Mounir Abou El Fattah.

[2] **OWPL**, *L'amélioration de la performance des processus et pratiques logiciels dans les petites entreprises françaises*, Anabel Strambolian, Avril 2006.

[3] **OWPL**, *Une Méthodologie et des Modèles Légers pour Initier une Démarche d'Amélioration des Pratiques Logicielles APL*, Naji Habra, Alain Renault.

[4] **Génie logciel**, *Ingénierie des logiciels*, Naji Habra, F.U.N.D.P.

[5] **Ingénierie des exigences**, *Requirements engineering*, Patrick Heymans, F.U.N.D.P.

[6] **Volere**, *Requirements Specification Template*, Edition 10.1, Atlantic Systems Guild.

[7] **Documentation**, *Documentation et Communication*, Czikus Carriere et Nadia Hijab, AMDD.

[8] **Gestion de projet**, *La gestion de projet*, http ://www.gestiondeprojet.net, C. Cornic.

# Rapport E :

*Horizon Vert - Cahier des charges*

# Classes d'utilisateurs

## Chef de projet

## Caractéristiques

1. **Attributs physiques :** La personne a l'âge légal pour exercer la fonction de chef de projet. Il n'y a à priori aucun âge spécifique ni de sexe typique. Rien n'est prévu au sein du logiciel pour la prise en charge de chef de projet à capacités réduites. Des incapacités physiques et handicaps modérés sont cependant tolérés.
2. **Attributs mentaux :** Le chef de projet est familiarisé aux concepts modernes des interfaces graphiques. Notre application n'est pas adaptée aux handicaps lourds. Du fait de sa qualification et de son intérêt évident pour la gestion des changements, le chef de projet adopte une attitude positive vis-à-vis de la tâche.
3. **Qualifications et connaissances :** La langue maternelle du chef de projet est le français. Les périphériques d'entrée utilisés sont le clavier et la souris.
4. **Caractéristiques du travail :** Le but du travail du chef de projet est de mettre à jour le système en gérant les demandes de changement ainsi qu'en gérant l'ajout et la suppression d'utilisateurs. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel.

## Environnement

1. **Localisation du produit :** Le logiciel est utilisé principalement dans les locaux de l'entreprise.
2. **Position :** Le chef de projet est généralement assis pour utiliser l'ordinateur et adopte vraisemblablement cette même position pour utiliser le logiciel.
3. **Matériel :** L'ordinateur du chef de projet est suffisamment puissant pour faire fonctionner la machine virtuelle java et notre application.
4. **Logiciel et Système d'exploitation :** Le système d'exploitation du chef de projet est Windows XP. Notre application ne requiert pas de système d'exploitation particulier mais une machine virtuelle Java doit être installée.
5. **Structure :**
   - *Travail en groupe :* Le logiciel est utilisé individuellement.
   - *Assistance :* Une aide incorporée au logiciel est disponible pour le chef de projet.
   - *Interruptions :* Ce facteur dépend fortement du contexte dans lequel le chef de projet se trouve. Dans le cas le plus probable, celui-ci n'est pas interrompu.
   - *Communications :* Le réseau de l'entreprise permet de véhiculer l'information liée à l'utilisation du logiciel.

## Utilisateur

## Caractéristiques

1. **Attributs physiques :** La personne a l'âge légal pour exercer la fonction d'utilisateur. Il n'y a à priori aucun âge spécifique ni de sexe typique. Rien n'est prévu au sein du logiciel pour la prise en charge d'utilisateurs à capacités réduites. Des incapacités physiques et handicaps modérés sont cependant tolérés.
2. **Attributs mentaux :** L'utilisateur est familiarisé aux concepts modernes des interfaces graphiques. Notre application n'est pas adaptée aux handicaps lourds. Du fait de sa qualification et de son intérêt évident pour la gestion des changements, l'utilisateur adopte une attitude positive vis-à-vis de la tâche.
3. **Qualifications et connaissances :** La langue maternelle de l'utilisateur est le français. Les périphériques d'entrée utilisés sont le clavier et la souris.
4. **Caractéristiques du travail :** Le but du travail de l'utilisateur est de mettre à jour le système en y incluant les nouvelles demandes de changement. Aucune contrainte quant à l'organisation du travail n'est imposée par le logiciel.

## Environnement

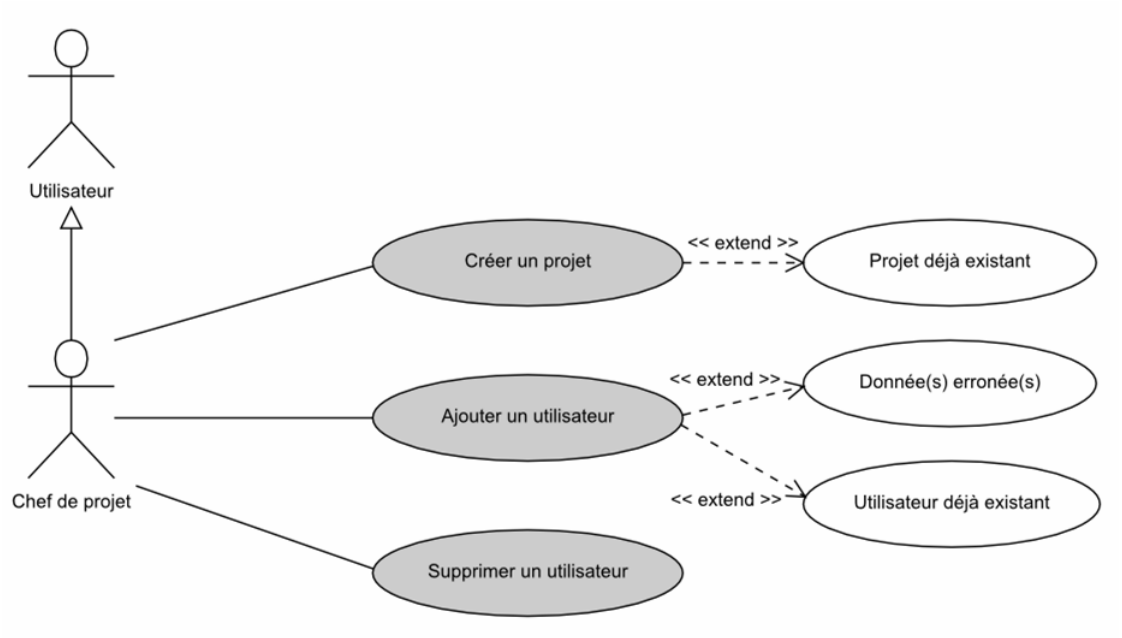
1. **Localisation du produit :** Le logiciel est utilisé principalement dans les locaux de l'entreprise.
2. **Position :** L'utilisateur est généralement assis pour utiliser l'ordinateur et adopte vraisemblablement cette même position pour utiliser le logiciel.
3. **Matériel :** L'ordinateur de l'utilisateur est suffisamment puissant pour faire fonctionner la machine virtuelle java et notre application.
4. **Logiciel et Système d'exploitation :** Le système d'exploitation du chef de projet est Windows XP. Notre application ne requiert pas de système d'exploitation particulier mais une machine virtuelle Java doit être installée.
5. **Structure :**
   - *Travail en groupe :* Le logiciel est utilisé individuellement.
   - *Assistance :* Une aide incorporée au logiciel est disponible pour l'utilisateur.
   - *Interruptions :* Ce facteur dépend fortement du contexte dans lequel l'utilisateur se trouve. Dans le cas le plus probable, celui-ci n'est pas interrompu.
   - *Communications :* Le réseau de l'entreprise permet de véhiculer l'information liée à l'utilisation du logiciel.

# Objectifs des utilisateurs

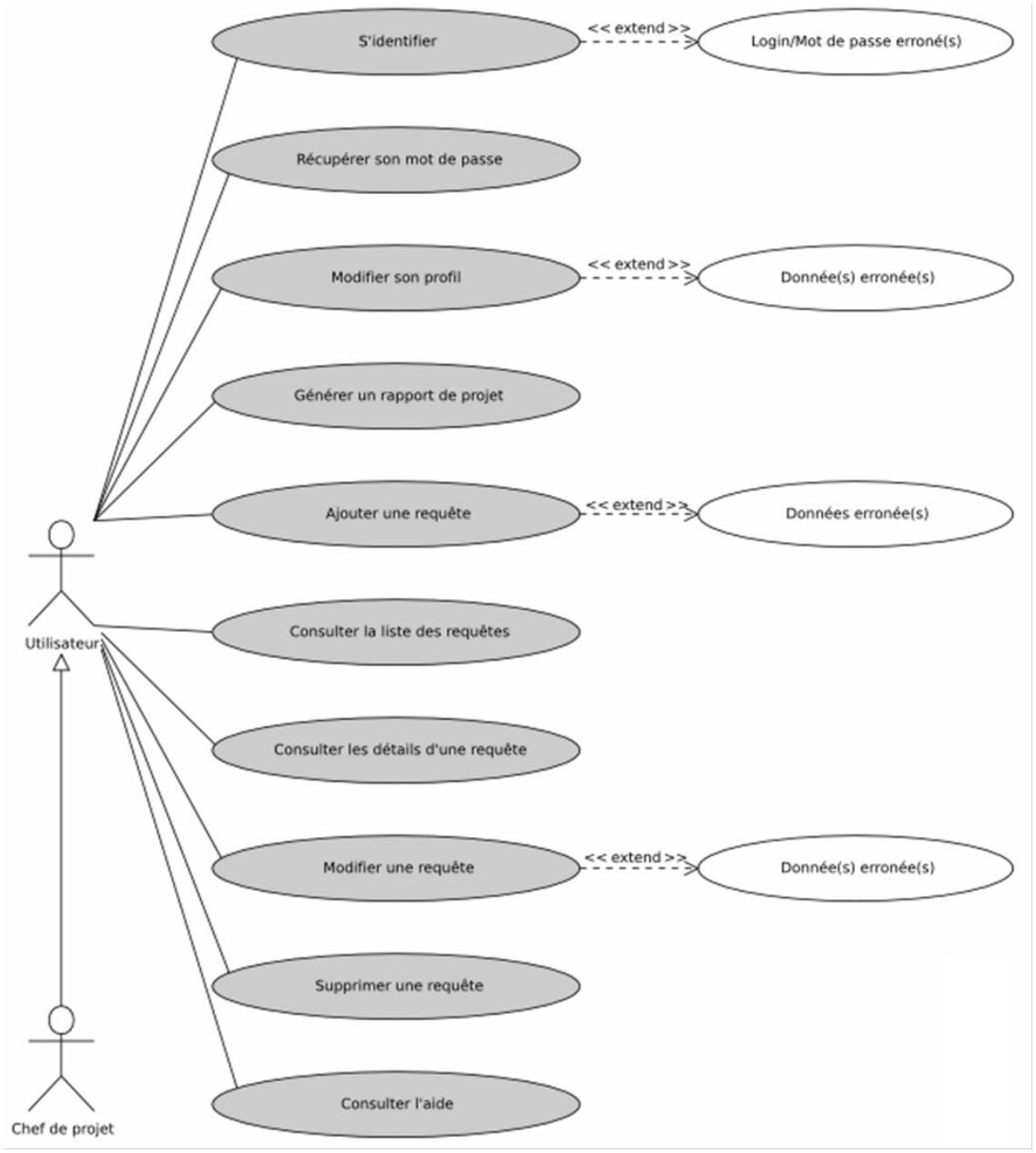
Les cas d'utilisation représentent le comportement affiché par le système sous certaines conditions de manière à satisfaire un objectif de l'un des acteurs. Les diagrammes de cas d'utilisation montrent quant à eux les acteurs, les limites du système, les relations entre acteurs et le système ainsi que les relations entre les cas d'utilisation eux-mêmes. Ceux-ci permettent, aux travers de scénarii, de capturer, représenter et valider les fonctions d'un domaine, les spécifier et donner un aperçu de la dynamique et des interactions. Du fait de leur simplicité, les cas d'utilisation ont pour avantage notable d'être compréhensibles par tous.

Notons que, afin d'avoir une vision plus précise et plus simple du système, nous l'avons divisé en deux parties : le système d'information du chef de projet (SI Chef de projet) et le système d'information de l'utilisateur (SI Utilisateur). De plus, toutes les fonctionnalités disponibles pour l'utilisateur le sont aussi pour le chef de projet.

## SI du chef de projet



## SI de l'utilisateur




Nous allons maintenant décrire les différents scénarii d'utilisation du chef de projet.

## Scénarii d'utilisation du chef de projet

## Créer un projet

### Résumé :

L'acteur crée un nouveau projet.

### Acteur :

Le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouveau projet est enregistré dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les caractéristiques du nouveau projet. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que le nouveau projet a bien été enregistré dans le système. |
| 5. L'acteur reçoit l'information. | |

## Extension : Projet déjà existant

### Résumé :

L'acteur crée un nouveau projet qui est déjà enregistré dans le système.

### Acteur :

Le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouveau projet est identique à un projet déjà enregistré dans le système.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouveau projet n'est pas enregistré dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les caractéristiques d'un projet existant. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que le projet existe déjà et lui demande de réintroduire les caractéristiques d'un nouveau projet. |
| 5. Retour au point 1 du cas normal. | |

## Ajouter un utilisateur

### Résumé :

L'acteur ajoute un nouvel utilisateur dans le système.

### Acteur :

Le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouvel utilisateur est enregistré dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les caractéristiques du nouvel utilisateur. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que le nouvel utilisateur a bien été enregistré dans le système avec un login déterminé. |
| 5. L'acteur reçoit l'information. | |

## Extension : Donnée(s) erronée(s)

### Résumé :

L'acteur ajoute un nouvel utilisateur dans le système avec une/des donnée(s) erronée(s).

### Acteur :

Le chef de projet.

### Précondition :

Le système est opérationnel.
L'acteur est identifié.

Une/des donnée(s) est/sont erronée(s).

### Postcondition :

Le système est opérationnel.
L'acteur est identifié.

Le nouvel utilisateur n'est pas enregistré dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les caractéristiques du nouvel utilisateur dont certaines sont erronées. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que certaines données sont erronées et lui demande de réintroduire correctement les caractéristiques d'un nouvel utilisateur. |
| 5. Retour au point 1 du cas normal. | |

## Extension : Utilisateur déjà existant

### Résumé :

L'acteur ajoute un nouvel utilisateur qui est déjà enregistré dans le système.

### Acteur :

Le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouvel utilisateur est identique à un utilisateur déjà enregistré dans le système.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le nouvel utilisateur n'est pas enregistré dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les caractéristiques d'un utilisateur déjà existant. | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que l'utilisateur existe déjà et lui demande de réintroduire les caractéristiques d'un nouvel utilisateur. |
| 5. Retour au point 1 du cas normal. | |

## Supprimer un utilisateur

### Résumé :

L'acteur supprime un des utilisateurs du système.

### Acteur :

Le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

L'utilisateur est enregistré dans le système.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

L'utilisateur n'est plus enregistré dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur sélectionne l'utilisateur à supprimer. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que l'utilisateur est bien supprimé. |
| 5. L'acteur reçoit l'information. | |

## Scénarii d'utilisation de l'utilisateur

## S'identifier

### Résumé :

L'acteur s'identifie afin d'accéder au programme de gestion des changements.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur lance le système.<br>2. L'acteur entre son login et son mot de passe.<br><br><br>5. L'acteur entre dans le programme. | <br><br><br>3. Le système reçoit les données.<br>4. Le système traite et vérifie les données. |

## Extension : Login et/ou mot de passe erroné(s)

### Résumé :

L'acteur s'identifie, afin d'accéder au programme de gestion des changements, avec un login et/ou un mot de passe erroné(s).

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur entre un login et/ou un mot de passe erroné(s).

### Postcondition :

Le système est opérationnel.

L'acteur n'est pas identifié.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur lance le système.<br>2. L'acteur entre son login et son mot de passe de façon erronée. | 3. Le système reçoit les données.<br>4. Le système traite et vérifie les données.<br>5. Le système signale à l'acteur qu'il a entré un login et/ou un mot de passe erroné(s) et lui demande de s'identifier à nouveau. |
| 6. Retour au point 2 du cas normal. | |

## Récupérer son mot de passe

### Résumé :

L'acteur tente de récupérer son mot de passe.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel.

Le mot de passe est envoyé par mail à l'utilisateur.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur lance le système.<br>2. L'acteur demande son mot de passe au système. | |
| | 3. Le système reçoit la demande.<br>4. Le système traite et vérifie la demande.<br>5. Le système envoie à l'utilisateur son mot de passe. |
| 6. L'utilisateur reçoit son mot de passe. | |

## Modifier son profil

### Résumé :

L'acteur modifie son profil.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le profil de l'acteur est modifié.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les donnés afin de modifier son profil. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que son profil a été modifié avec succès. |
| 5. L'acteur reçoit l'information. | |

## Extension : Donnée(s) erronée(s).

### Résumé :

L'acteur modifie son profil avec une/des donnée(s) erronée(s).

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

Certaines données entrées par l'acteur sont erronées.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le profil de l'acteur est inchangé.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre une/des donnés erronée(s) afin de modifier son profil. | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système demande à l'acteur de rentrer des données correctes. |
| 5. Retour au point 1 du cas normal. | |

## Générer un rapport de projet

### Résumé :

L'acteur génère, au format pdf, le rapport des requêtes de changement relatives à un projet.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Le rapport est généré.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur demande au système de générer le rapport relatif à un projet. | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que le rapport a été créé et placé dans le répertoire relatif au projet concerné. |
| 5. L'acteur reçoit l'information. | |

## Ajouter une requête

### Résumé :

L'acteur ajoute une nouvelle requête de changement dans le système.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

La nouvelle requête de changement est enregistrée dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les caractéristiques concernant la nouvelle requête de changement. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que la nouvelle requête de changement a bien été enregistrée dans le système. |
| 5. L'acteur reçoit l'information. | |

## Extension : Donnée(s) erronée(s)

### Résumé :

L'acteur ajoute une nouvelle requête de changement dans le système avec une/des donnée(s) erronée(s).

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

Une/des donnée(s) est/sont erronée(s).

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

La nouvele requête de changement n'est pas enregistrée dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les caractéristiques de la nouvelle requête de changement avec une/des donnée(s) erronée(s). | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système demande à l'acteur de rentrer des données correctes. |
| 5. Retour au point 1 du cas normal. | |

## Consulter la liste des requêtes

### Résumé :

L'utilisateur consulte la liste des requêtes de changement d'un projet particulier et selon un ordre spécifié.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

L'utilisateur a accès à la liste des requêtes de changement d'un projet particulier et selon un ordre spécifié.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'utilisateur demande pour consulter la liste des requêtes de changement d'un projet particulier et selon un ordre bien précis. | |
| | 2. Le système reçoit la demande.<br>3. Le système traite la demande.<br>4. Le système envoie la liste des requêtes du projet, classée selon l'ordre spécifié. |
| 5. L'acteur a accès à la liste des requêtes du projet selon l'ordre spécifié. | |

## Consulter les détails d'une requête

### Résumé :

L'acteur consulte les détails d'une requête, à savoir, l'ensemble de ses caractéristiques ainsi que la liste des requêtes liées (requêtes qui référencent au moins une des exigences référencées par la requête consultée).

### Acteur :

Le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

L'acteur reçoit les détails de la requête.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur spécifie la requête pour laquelle il veut obtenir les détails, et ce, en consultant la liste des requêtes. | |
| | 2. Le système reçoit la demande.<br>3. Le système traite la demande.<br>4. Le système envoie les détails de la requête spécifiée. |
| 5. L'acteur a accès aux détails de la requête spécifiée. | |

## Modifier une requête

### Résumé :

L'acteur modifie certaines caractéristiques d'une requête de changement enregistrée dans le système.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

La requête de changement est enregistrée dans le système.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Certaines caractéristiques de la requête de changement sont changées.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre les nouvelles caractéristiques de la requête de changement. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que les changements ont été effectués. |
| 5. L'acteur reçoit l'information. | |

## Extension : Donnée(s) erronée(s)

### Résumé :

L'acteur modifie, de façon erronée, certaines caractéristiques d'une requête de changement enregistrée dans le système.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

La requête de changement est enregistrée dans le système.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

Les caractéristiques de la requête de changement sont inchangées.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur entre, de façon erronée, les nouvelles caractéristiques de la requête de changement. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système demande à l'acteur de rentrer des données correctes. |
| 5. Retour au point 1 du cas normal. | |

## Supprimer une requête

### Résumé :

L'acteur supprime une des requêtes de changement enregistrée dans système.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

L'acteur est identifié.

La requête de changement est enregistrée dans le système.

### Postcondition :

Le système est opérationnel.

L'acteur est identifié.

La requête de changement n'est plus enregistrée dans le système.

### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'acteur sélectionne la requête de changement à supprimer. | |
| | 2. Le système reçoit les données.<br>3. Le système traite et vérifie les données.<br>4. Le système avertit l'acteur que la requête de changement est supprimée. |
| 5. L'acteur reçoit l'information. | |

## Consulter l'aide

### Résumé :

L'utilisateur demande de l'aide sur la manière dont fonctionne le programme.

### Acteur :

L'utilisateur et le chef de projet.

### Précondition :

Le système est opérationnel.

### Postcondition :

Le système est opérationnel.

L'utilisateur reçoit l'aide demandée.

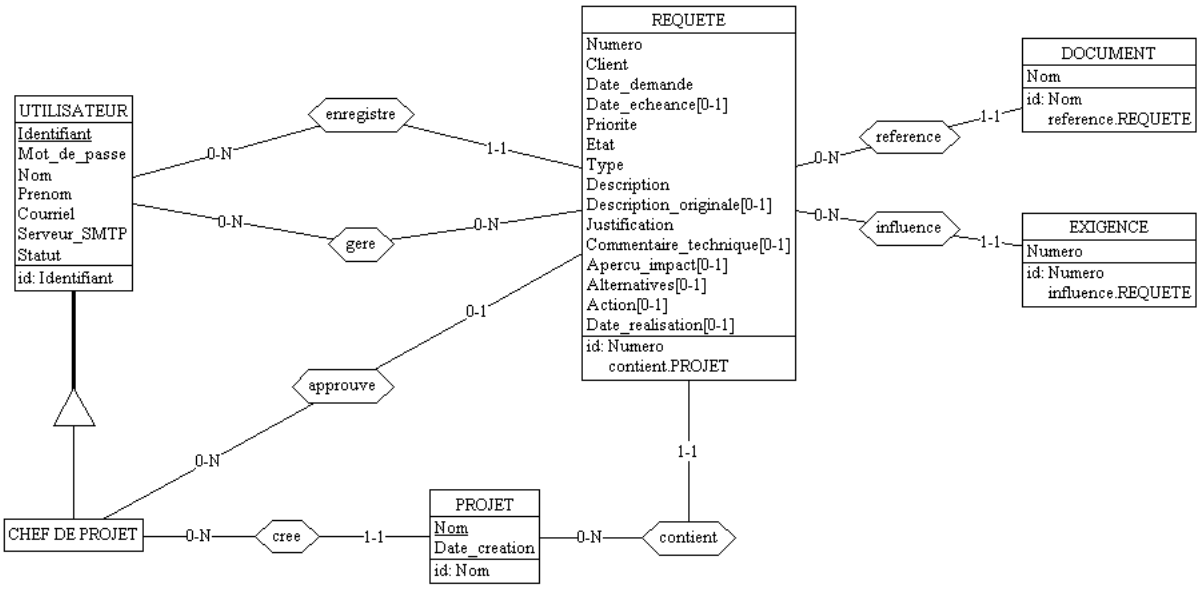
### Flux d'événement :

| *Acteur* | *Réponse du système* |
|---|---|
| 1. L'utilisateur demande de l'aide sur la manière dont fonctionne le programme. | |
| | 2. Le système reçoit la demande. |
| | 3. Le système traite la demande. |
| | 4. Le système envoie l'aide à l'acteur. |
| 5. L'acteur reçoit l'aide demandée. | |

# Diagramme de la statique

## Schéma de la statique

Le modèle Entité-Relation-Attribut propose des concepts (principalement les entités, les associations et les attributs) permettant de décrire un ensemble de données relatives à un domaine défini.

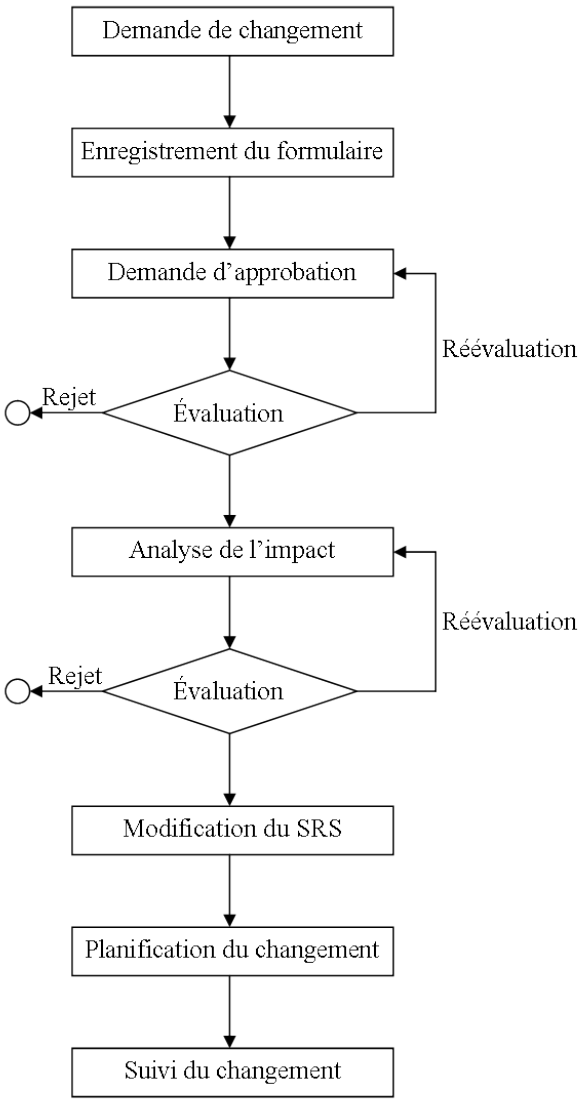


# Rapport F :

*Horizon Vert - Cahier d'implémentation*

# Processus de gestion des changements

Le processus de gestion des changements, représenté à la figure VII, a pour but d'aider l'entreprise à gérer formellement les requêtes de changement provenant de leurs clients.



## Demande de changement

Cette première étape consiste en une requête de la part du client qui souhaite voir un changement dans le logiciel qu'il utilise. Pour ce faire, le client contacte l'entreprise via l'un des points de contact, préalablement établi par cette entreprise pour permettre aux clients d'émettre leurs requêtes. Ces points de contact peuvent être un numéro de téléphone, un courriel, une adresse, etc.

## Enregistrement du formulaire

Cette étape représente l'enregistrement de la requête de changement via un formulaire bien défini. La personne en charge du point de contact va réaliser et enregistrer une description détaillée et formelle du changement voulu par le client. Remarquons que le modèle de ce formulaire est décrit dans le chapitre suivant.

À la fin de cette étape, une requête de changement est enregistrée et est dans l'état « ENREGISTRÉE ».

## Demande de validation

Cette première évaluation va permettre de décider s'il y a lieu ou non de réaliser la requête de changement. C'est le chef de projet qui est responsable d'évaluer la faisabilité du changement et de recontacter éventuellement le client pour de plus amples informations.

À la fin de cette étape, la requête de changement est dans l'état « VALIDÉE » ou « REJETÉE ».

## Analyse de l'impact

Cette seconde évaluation va permettre de décider s'il y a lieu ou non de réaliser la requête de changement, préalablement validée, en fonction de son impact sur le système existant. Le chef de projet va donc étudier l'impact que le changement aurait sur les autres exigences et prendre en considération toutes les alternatives possibles à ce changement.

À la fin de cette étape, la requête de changement est dans l'état « APPROUVÉE » ou « REJETÉE ».

## Modification du SRS

À cette étape, il s'agit de traduire la requête de changement, préalablement approuvée, dans la spécification des exigences existantes et ce, via l'outil GenSpec. Un changement dans le système peut impliquer soit, une nouvelle exigence pour le système, soit une modification d'une ou plusieurs exigences existantes.

À la fin de cette étape, la requête de changement est dans l'état « SPÉCIFIÉE ».

## Planification du changement

Une fois le changement spécifié, il s'agit de planifier la réalisation de celui-ci. Le chef de projet est donc responsable de répartir, dans le temps, les tâches nécessaires à la réalisation du changement entre les membres de son équipe.

À la fin de cette étape, la requête de changement est dans l'état « PLANIFIÉE ».

## Réalisation du changement

Lors de cette étape, les ressources allouées au développement du changement vont réaliser ce dernier en respectant la planification préétablie.

À la fin de cette étape, la requête de changement est dans l'état « RÉALISÉE ».

## Suivi du changement

Cette dernière étape consiste en un suivi de la réalisation du changement qui a été planifié. Pour ce faire, le chef de projet consulte régulièrement les ressources allouées afin de connaître l'avancement du développement et constitue, ensuite, une équipe de testeurs qui se chargera de réaliser des plans de tests bien définis afin de vérifier la modification apportée au système.

À la fin de cette étape, la requête de changement est dans l'état « TESTÉE ».

# Formulaire de changement

Ce chapitre décrit le modèle de formulaire utilisé pour l'enregistrement des requêtes de changement provenant des clients. Ce modèle est composé de trois parties complémentaires, à savoir, l'*identification*, la *description* et l'*analyse*.

Les parties concernant l'*identification* et la *description* de la demande, seront enregistrées par la personne en charge du point de contact avec le client. Les trois parties pourront ensuite être mises à jour par le chef de projet, au fur et à mesure de la progression de la demande à travers les étapes du processus. Remarquons que les éléments dotés d'une astérisque sont obligatoires.

## Identification

| |
|---|
| **Nom du projet* :** *Le nom du produit à développer.* |
| **Numéro de la requête* :** *Le numéro de la requête dans le projet.* |
| **Titre* :** *Le titre résumant le but de la requête.* |
| **Émise par* :** *Le client qui émet la requête.* |
| **Enregistrée par* :** *Le point de contact du client, celui qui enregistre la requête.* |
| **Priorité* :** *Un niveau de priorité de la requête basé sur trois niveaux.* |
| **État* :** *L'état de la requête, tel que précisé dans le processus de gestion des changements.* |
| **Date de demande* :** *La date à laquelle la requête a été émise par le client.* |
| **Date d'échéance :** *La date à laquelle la requête devient inutile pour le client.* |

## Description

| |
|---|
| **Type* :** *Le type de la requête, tel que : Ajout, Modification, Retrait,...* |
| **Description* :** *Une description de la requête, de ce que le client voudrait.* |
| **Description originale :** *Une description de l'existant, i.e, comment cela est géré pour le moment.* |
| **Justification* :** *La raison qui justifie la réalisation de la requête.* |
| **Commentaire technique :** *Un commentaire technique sur la requête.* |
| **Documents relatifs :** *Des documents annexes permettant d'illustrer la requête.* |

## Analyse

## Évaluation

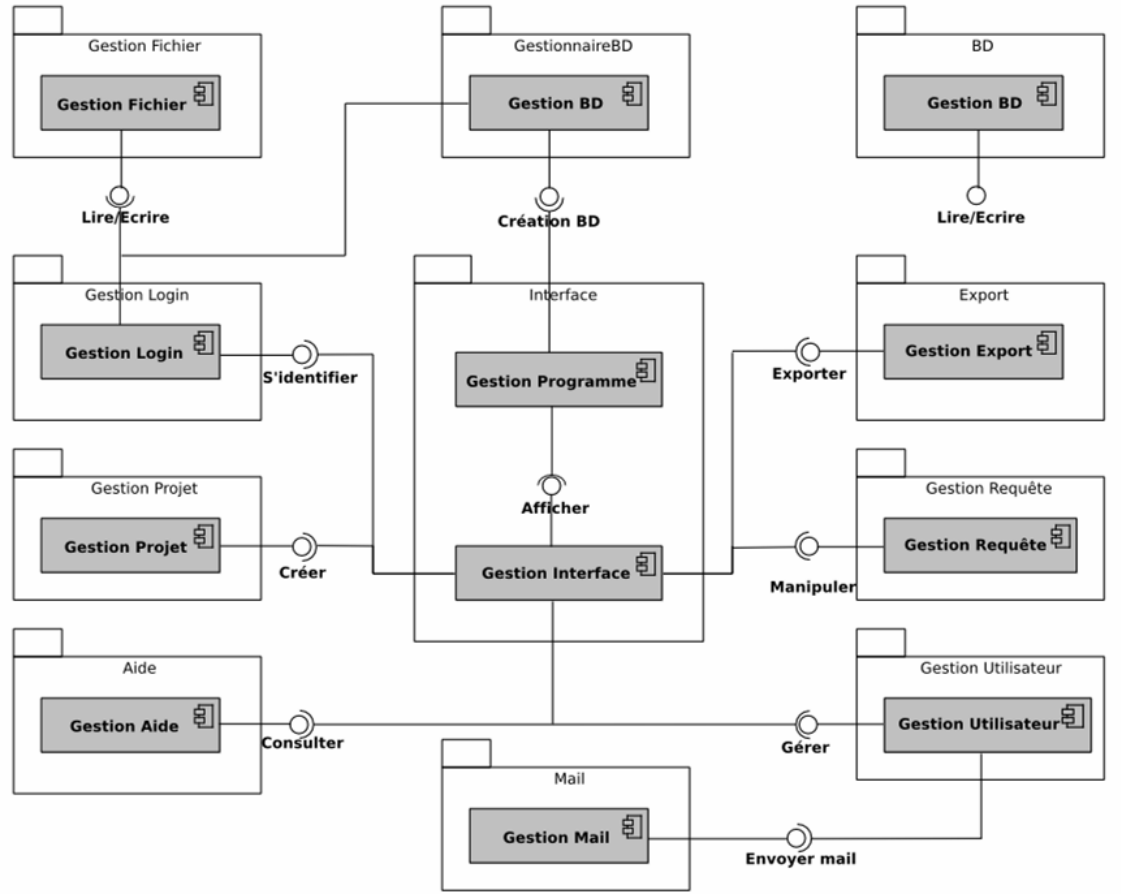
| |
|---|
| **Aperçu de l'impact* :** *Description de l'impact du changement sur les autres exigences.* |
| **Exigences concernées :** *La liste des numéros d'exigences qui sont concernées par le changement demandé.* |
| **Alternatives :** *Les alternatives possibles relatives au changement demandé.* |
| **Action* :** *Ce qu'il convient d'entreprendre pour réaliser le changement : Temps, Ressources,...* |

## Planification

| |
|---|
| **Analyste* :** *La personne responsable d'évaluer et de planifier le changement.* |
| **Approbateur* :** *La personne qui donne son accord pour réaliser le changement.* |
| **Implémenteur* :** *La personne responsable d'implémenter le changement.* |
| **Date d'approbation :** *La date à laquelle l'approbateur a donné son accord.* |
| **Date de début des travaux :** *La date planifiée de début des travaux.* |
| **Date de fin des travaux :** *La date planifiée de fin des travaux.* |
| **Date finale :** *La date de réalisation finale du changement.* |

# Diagramme de composants

Le diagramme de composants décrit l'organisation du système du point de vue des modules de code. Ce diagramme permet de mettre en évidence les dépendances entre les composants (qui utilise quoi) et ainsi permet de mieux organiser les modules. L'encapsulation (insérer un composant dans un autre, cacher des détails d'implémentation,...) permet de réduire la complexité et d'élever le niveau d'abstraction. Le système développé pour Horizon Vert peut donc être représenté par le diagramme de composants suivant :




Le digramme de composants représenté par la figure ci-dessus peut être décrit en 3 parties que voici :

## Gestion du programme

Le composant "Gestion Programme" a pour fonction de lancer le gestionnaire de requêtes de changement. Une fois sa tâche terminée, il passe la main soit au composant "GestionnaireBD.Gestion BD" s'il ne trouve pas les informations nécessaires pour se connecter à la base de données, ou au composant "Gestion Interface" dans le cas contraire.

## Gestion de la base de données

Lors du premier lancement du gestionnaire, il est nécessaire de passer par une phase d'installation permettant d'obtenir les informations nécessaires pour se connecter à la base de données, d'y insérer les tables si elles n'y existent pas encore et de créer son propre compte pour s'identifier au gestionnaire. Les composants nécessaires à cette première phase sont :

- **GestionnaireBD.GestionBD :** Ce composant contient l'ensemble des fonctionnalités nécessaires pour mener à bien la phase d'installation du sytème.
- **Gestion Fichier :** Il permet de sauvegarder les informations nécessaires pour se connecter à la base de données lors de chaque lancement du gestionnaire. Ce composant offre des méthodes permettant d'écrire et de lire dans un fichier créé préalablement par le gestionnaire.

Une fois la phase d'installation terminée, la main est rendue au composant "Gestion Programme" qui va ensuite appeler le composant "Gestion Interface" pour afficher le gestionnaire de requêtes de changement.

## Gestion des interactions

Le composant "Gestion Interface" constitue le composant principal du système. En effet, selon l'interaction de l'utilisateur, il va passer la main à un autre composant qui va réaliser la tâche demandée et ensuite la lui redonner. Tous ces composants interagissent avec la base de données via le composant "BD"[12]. Plusieurs cas sont possibles :

*Gestion Login*

Ce composant s'occupe de l'identification de l'utilisateur lors du lancement de modeX et de la récupération du mot de passe. Il interagit avec le composant "Gestion Fichier" afin de récupérer les informations nécessaires pour se connecter à la base de données et ainsi, vérifier si l'utilisateur est bien la personne qu'il prétend être[13].

[12] Aucun lien n'a été tracé vers ce composant afin de ne pas alourdir le diagramme

[13] Le fichier contient ces informations puisqu'une phase d'installation a été réalisée lors du premier lancement de modeX.

*Gestion Projet*

Ce composant s'occupe de créer les projets que le chef de projet spécifie. Une arborescence est alors créée pour chaque projet afin d'y enregistrer les rapports d'exportation.

*Gestion Aide*

Ce composant s'occupe d'afficher à l'utilisateur le fichier d'aide livré avec le gestionnaire de requêtes de changement modeX.

*Gestion Export*

Ce composant s'occupe d'exporter le rapport d'un projet déterminé selon les différentes caractéristiques fournies par l'utilisateur (Parties à intégrer, ordre de classement,...).

*Gestion Requête*

Ce composant contient les fonctions d'ajout d'une nouvelle requête, de modification et de suppression d'une requête déjà existante ainsi que de l'affichage des détails et exigences dépendantes d'une requête sélectionnée préalablement.

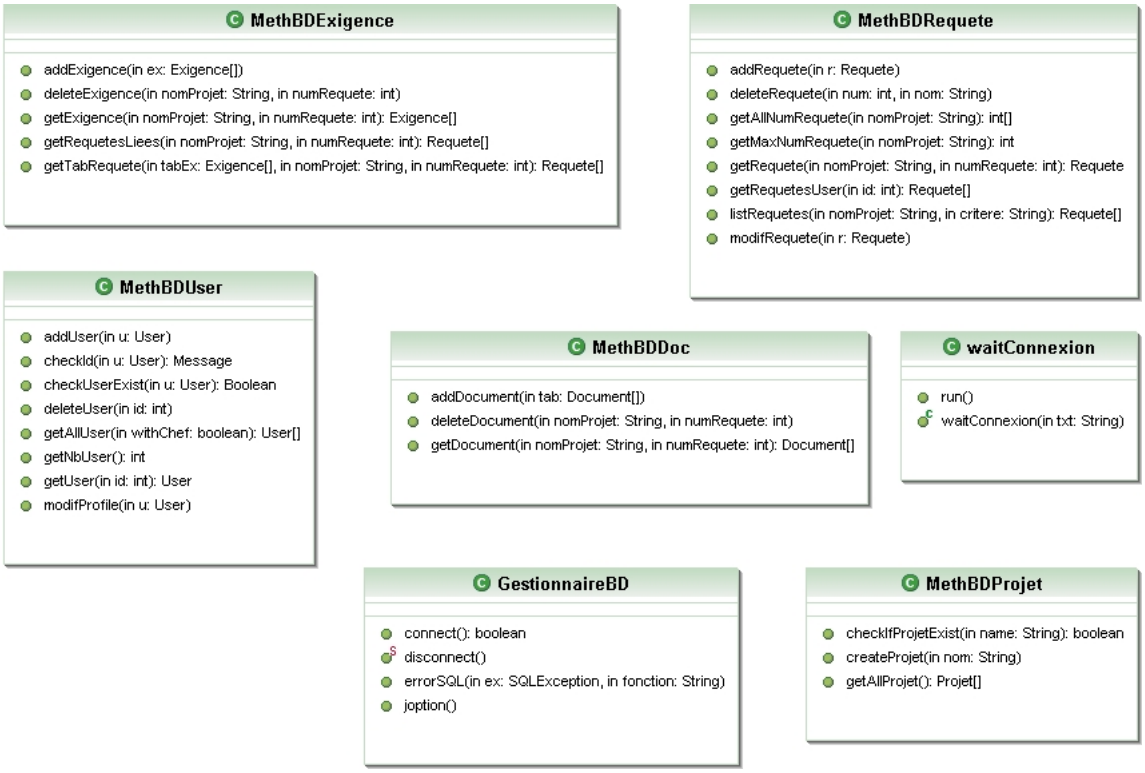
*Gestion Utilisateur*

Ce composant gère les fonctionnalités relatives à la gestion des utilisateurs, à savoir : ajouter et supprimer un utilisateur. Il travaille avec le composant "Gestion Mail" afin d'avertir l'utilisateur de la création ou de la suppression de son compte.

# Diagramme de classes

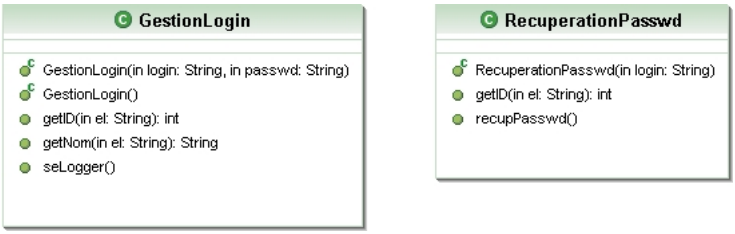
## Package BD

Ce package contient l'ensemble des méthodes nécessaires pour intéragir avec la base de données (enregistrer, modifier, supprimer, récupérer,... des données).



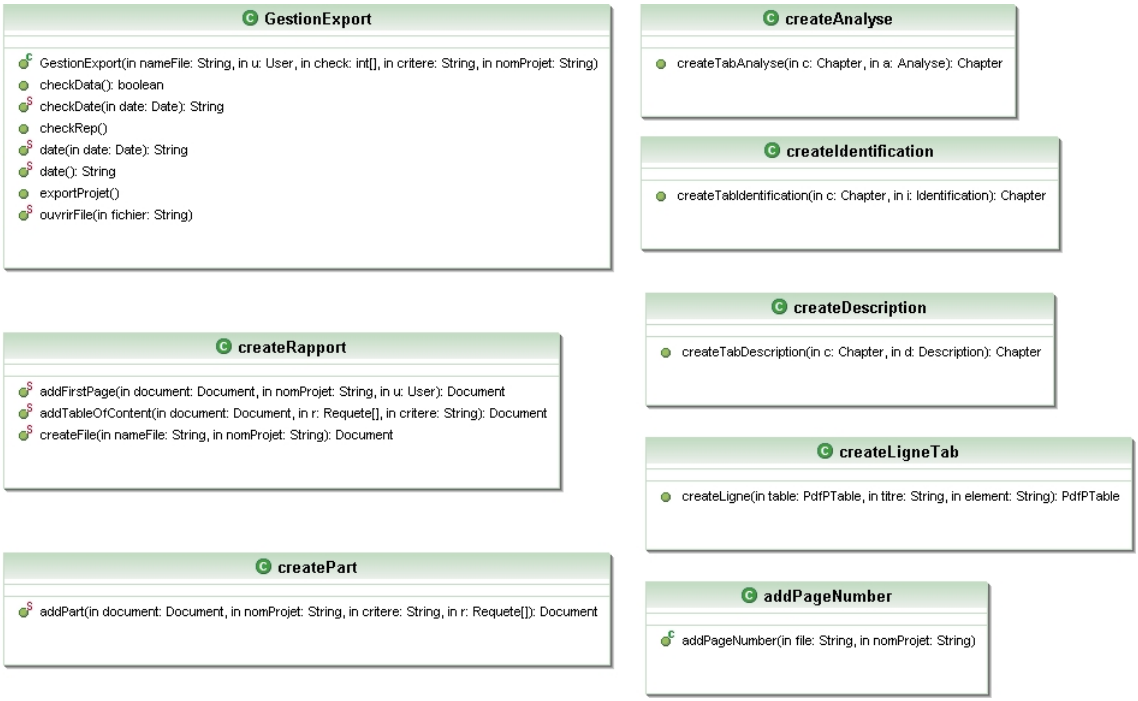
## Package GestionLogin

Ce package gère l'identification de l'utilisateur lors du lancement du programme ainsi que la récupération du mot de passe.



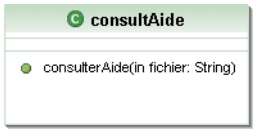

## Package Export

Ce package comprend l'ensemble des fonctions utilisées pour exporter les rapports de projet au format pdf.



## Package GestionAide

Ce package contient les méthodes nécessaires afin d'afficher le fichier d'aide lors de la demande de l'utilisateur.

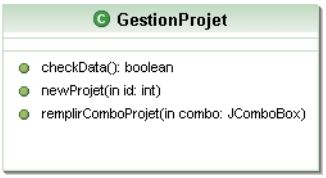



## Package GestionFichier

Ce package contient l'ensemble des méthodes permettant d'interagir avec le fichier où sont enregistrés les différentes caractéristiques nécessaires pour se connecter à la base de données.

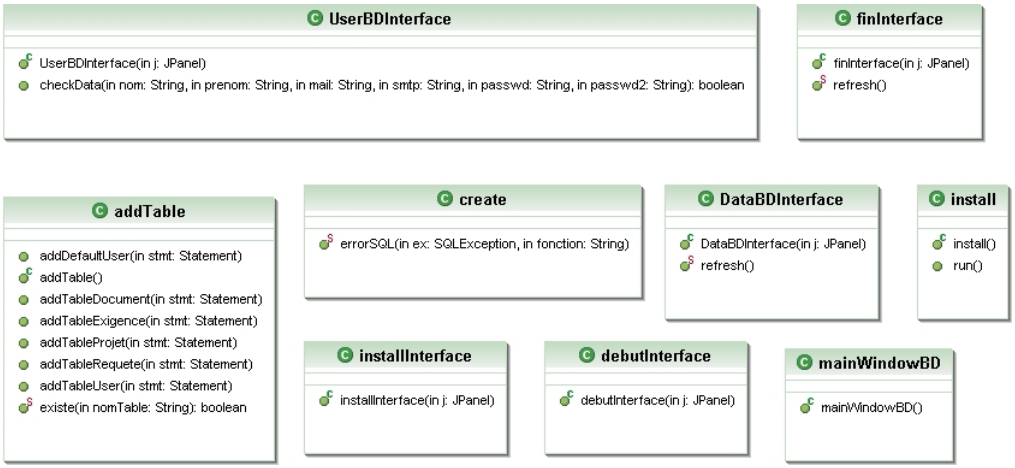

## Package GestionProjet

Ce package contient les méthodes nécessaires pour créer un nouveau projet ainsi que l'arborescence qui lui est associée.



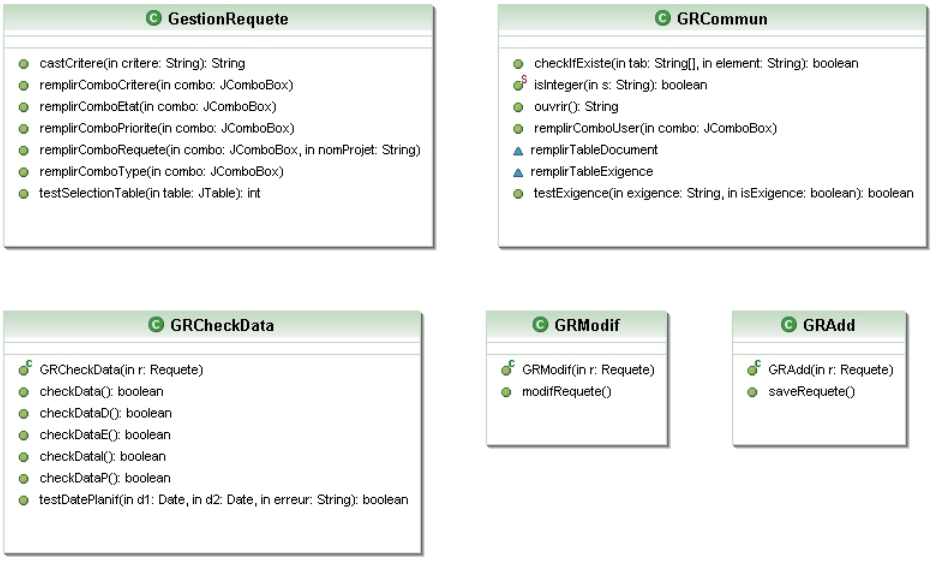
## Package GestionnaireBD

Ce package contient l'ensemble des méthodes nécessaires lors de la phase d'installation du programme sur un nouvel ordinateur.



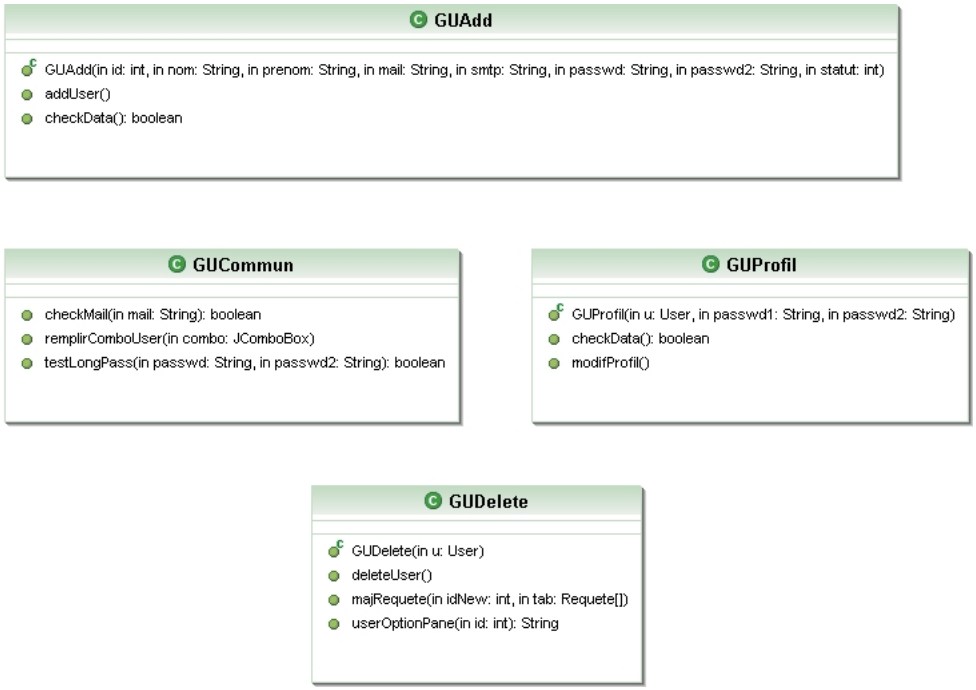

## Package GestionRequete

Ce package contient l'ensemble des méthodes nécessaires à la bonne gestion des requêtes de changement, à savoir : ajouter, modifier, supprimer et lister les détails d'une requête.



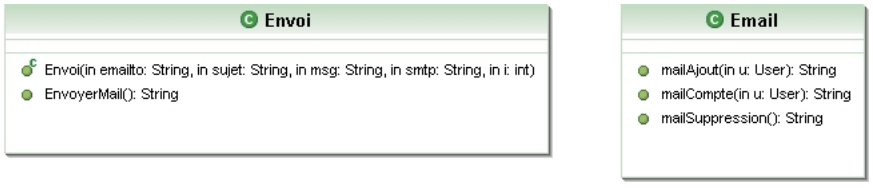
## Package GestionUtilisateurs

Ce package contient l'ensemble des méthodes nécessaires à la bonne gestion des utilisateurs, à savoir : ajouter ou supprimer un utilisateur.




## Package Mail

Ce package contient l'ensemble des méthodes nécessaires pour élaborer et envoyer un e-mail lorsqu'une fonctionnalité le requière.

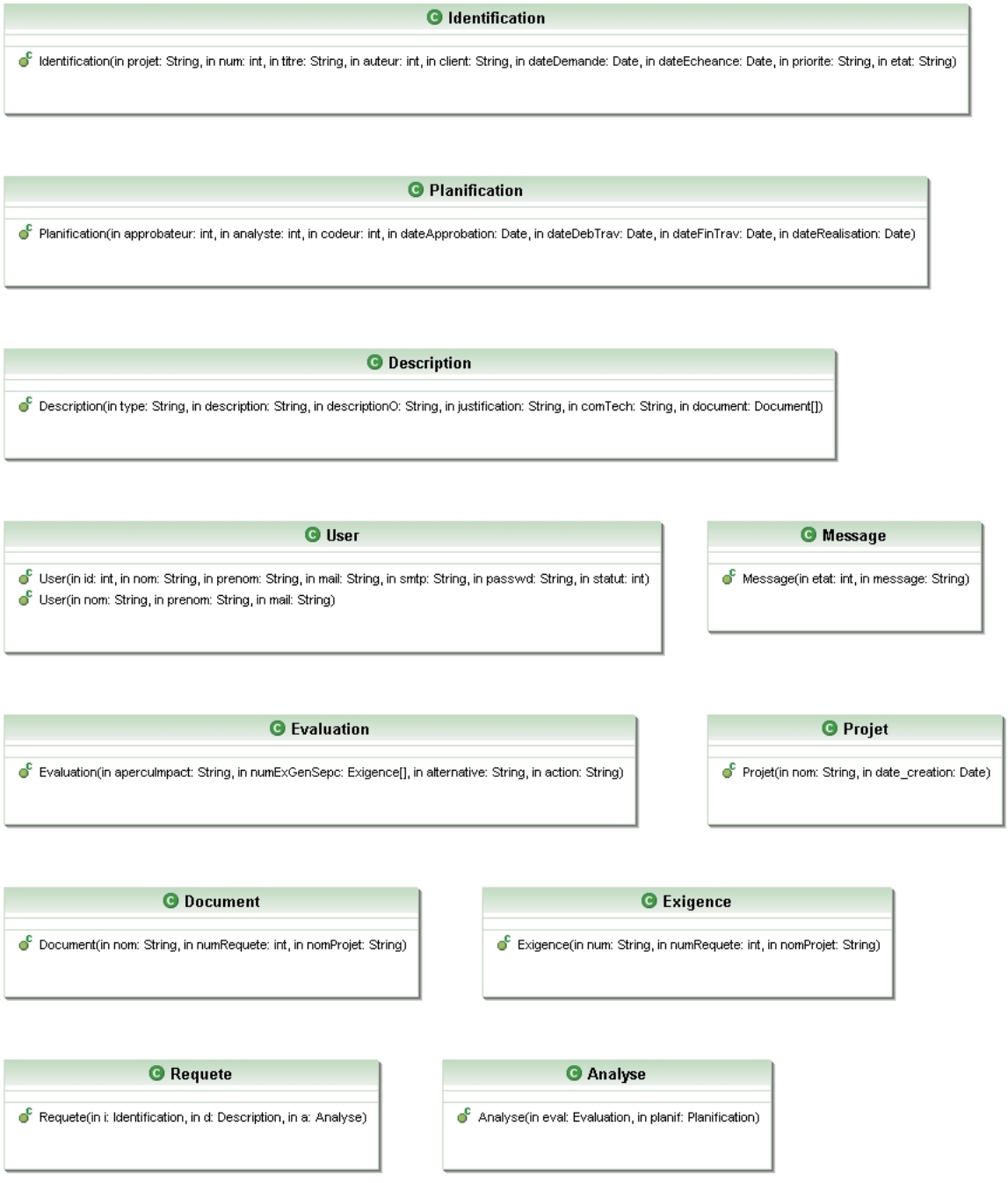


## Package Interface

Ce package contient l'ensemble des méthodes nécessaires pour afficher le programme à l'utilisateur et ainsi, lui proposer un ensemble de fonctionnalités.

## Package Objet

Ce package contient l'ensemble des objets java utilisés dans le programme.



# Rapport G :

*Horizon Vert - Présentation de GenSpec*

# L'ingénierie des exigences

A travers cette première partie, nous voulons mettre en avant les apports relatifs à une bonne gestion des exigences et ainsi, conforter la nécessité de notre intervention chez Horizon Vert.

## Son contexte

L'ingénierie des exigences (ou IE) est une activité du processus de fourniture et d'acquisition. Elle constitue un lien entre le client et le fournisseur. Ses intrants sont les besoins ou exigences brutes spécifiés par le client. Ses extrants sont les documents d'exigences : norme, appel d'offres, contrat, devis, cahier des charges, spécification, etc...

Les exigences sont considérées comme des pré-requis pour les étapes de conception et de développement d'un produit. La phase de développement des exigences peut avoir été précédée par une étude de faisabilité qui permet, comme son nom l'indique, de vérifier que le projet est faisable par l'entreprise en question.

## Son contenu

L'ingénierie des exigences inclut les quatre points suivants :

1. La compréhension du domaine du problème, l'identification des parties prenantes, l'identification des buts des parties prenantes, l'identification des contraintes de développement, l'identification des risques du projet et la documentation des exigences ;
2. La négociation des exigences entre le client et le fournisseur ;
3. l'implantation et le suivi de la traçabilité des exigences ;
4. la gestion des modifications d'exigences.

## Son importance

L'IE est une activité très importante du processus de fourniture et d'acquisition. À tel point que, si elle est négligée, plusieurs besoins du client ne sont jamais compris par le fournisseur

ou ne le sont qu'après ou peu avant la livraison. Il en découle les problèmes majeurs suivants :

- **Augmentation des coûts et délais de réalisation :** la compréhension d'un besoin après ou peu avant la livraison implique souvent de recommencer la réalisation, au moins en partie ;
- **Diminution de la qualité :** l'incompréhension d'un besoin implique que le produit (ou service) ne répondra pas à ce besoin ; et la compréhension d'un besoin après ou peu avant la livraison implique souvent que le produit ne répondra pas à ce besoin ou ne sera que sommairement corrigé pour y répondre le mieux possible.
- L'IE est une activité non seulement importante mais aussi essentielle à la fourniture et à l'acquisition. En effet, les exigences sont la base de l'entente client-fournisseur. De surcroît, elles sont la base de la fourniture et de l'acquisition : base de réalisation ; base de validation et d'acceptation par le client ; base de documentation.

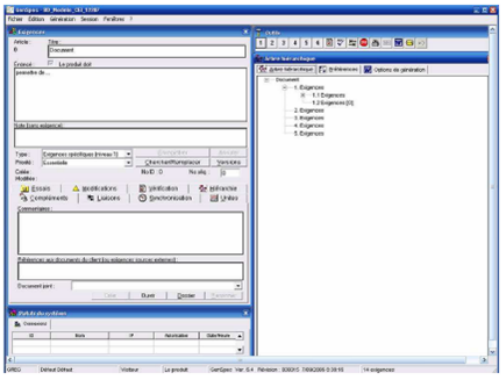
Il est donc nécessaire, afin de bien mener ses projets, de passer par une phase de spécification et de formalisation des exigences. Cependant, cette phase est coûteuse et difficile à mettre en oeuvre.

# Notre solution : GenSpec

Pour solutionner le problème de gestion des exigences chez Horizon Vert, nous avons opté comme solution le logiciel GenSpec d'Hydro-Québec. Ses avantages sont que c'est un outil gratuit, utilisable sous l'environnement Windows et compatible avec le template Volere qui offre des cartes de descriptions des exigences.

## Description générale

GenSpec permet l'entrée des exigences dans une base de données, plusieurs vérifications automatiques de ces exigences et la génération de documents d'exigences. La figure 2.1 présente son interface homme-machine : à droite apparaît l'arbre d'exigences ; à gauche, le formulaire d'entrée de l'exigence sélectionnée.



GenSpec a été développé notamment à partir de normes internationales, de documents de la NASA et du Département de la Défense des États-Unis d'Amérique. On obtient ainsi :

- Des exigences hiérarchisées ;
- Une seule exigence par paragraphe ;
- Un numéro de référence unique par exigence,...

En particulier, la norme 12207 de ISO/CEI/IEEE, un document de très haute qualité, a été utilisée comme « modèle » de document d'exigences

## Fonctions principales

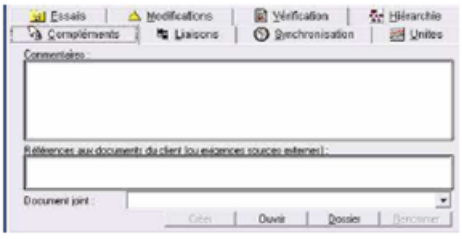
## Définition des exigences

Cette fonction permet à plusieurs utilisateurs en même temps (multi-utilisateurs) d'entrer ou de modifier des exigences ; elle supporte notamment les commandes Chercher et Remplacer (voir figure 2.2). De plus, elle permet de générer les documents d'exigences : spécification, arbre d'exigences,... (voir figure 2.3).



## Caractérisation des exigences

Par exigence, cette fonction génère un numéro de référence unique et permet d'entrer l'identification de la source (référence à un paragraphe d'un autre document), la priorité, une note, un commentaire et un fichier joint, tous pouvant être générés dans le document d'exigences (voir figure 2.4)[14].




[14] A la demande, les exigences commentées apparaissent en vert dans l'arbre d'exigences.

## Structuration et liaison des exigences

Cette fonction permet de structurer et de lier les exigences (renvois) par de simples commandes clic et glisse. De plus, elle offre des facilités de navigation telle qu'une commande d'aller-retour rapide entre l'origine et la destination d'un lien[15].

## Évaluation de conformité aux exigences

Par exigence, cette fonction permet d'entrer des procédures d'évaluation de conformité, et le résultat de cette évaluation (voir figure 2.5). De plus, elle permet de générer un rapport d'évaluation contenant les exigences, leurs procédures d'évaluation et les résultats de cette évaluation[16].


## Fonctions secondaires

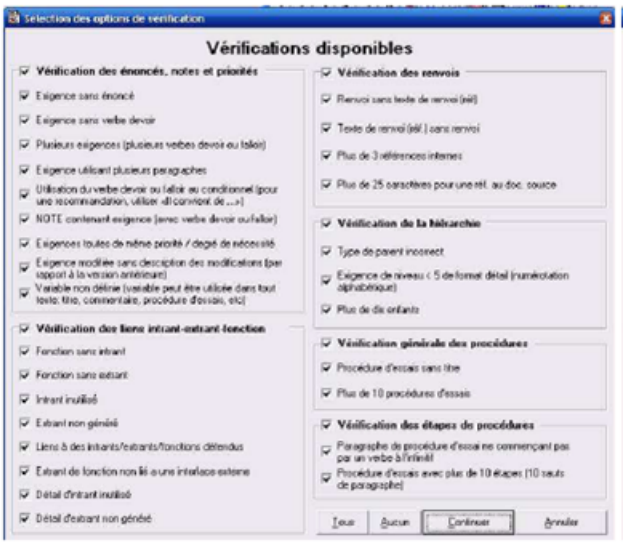
## Contrôle et analyse des exigences

Cette fonction empêche d'introduire des incohérences de hiérarchie ou de liaison d'exigences - règles de hiérarchie et de liaison paramétrables - et, en particulier, de supprimer une exigence à laquelle d'autres renvoient. De plus, elle offre un vérificateur d'exigences (voir figure 2.6), y compris un vérificateur d'orthographe et de grammaire. De surcroît, elle permet de générer un tableau Sources Vs Exigences, facilitant la vérification de l'exactitude des exigences[17].

[15] A la demande, les exigences reliées apparaissent en bleu dans l'arbre d'exigences.

[16] En double-cliquant dans « Procédure », l'utilisateur a accès à une plus grande zone d'édition.

[17] Les erreurs détectées sont identifiées sous l'onglet « Vérification » de la fenêtre « Exigences ». De plus, à la demande, les exigences en erreur apparaissent en rouge dans l'arbre d'exigences.




## Normalisation des exigences

Cette fonction permet de générer automatiquement des textes de début d'exigence selon le type d'exigence - texte et type paramétrables. De plus, elle permet de définir et d'utiliser des variables dans les textes d'exigences.

## Configuration des documents d'exigences

Cette fonction offre une grande quantité d'options de formatage des documents générés ; elle permet notamment d'en exclure des exigences et d'en inclure d'autres avec la mention « Non applicable ».

## Gestion de l'historique des exigences

Cette fonction permet d'entrer et de visualiser la raison de modification d'une exigence par rapport à la version antérieure, et d'enregistrer une version formelle de l'ensemble des exigences. De plus, elle permet de comparer la version actuelle avec une version antérieure, et de ramener une ou toutes les exigences telles qu'elles étaient à une version antérieure. Enfin, elle permet de générer un tableau Historique des modifications d'exigence. En complément, elle permet d'enregistrer de simples copies de sécurité des exigences.

# Avantages de GenSpec

Cette section présente les avantages de GenSpec par rapport à un logiciel de traitement de texte tel que Word ou openOffice.

## Réduction des coûts

GenSpec réduit les coûts de l'ingénierie des exigences :

- concentre les efforts sur les exigences plutôt que sur le formatage, les documents étant générés automatiquement ;
- permet de structurer facilement les exigences ;
- permet de générer automatiquement des textes de début d'exigence, aidant, de surcroît, à la normalisation ;
- facilite la liaison des exigences ;
- génère automatiquement le rapport d'évaluation ;
- facilite la lecture, réduit la quantité d'erreurs et respecte des normes internationales.

## Facilité de lecture

GenSpec facilite la lecture des exigences :

- Oriente à bien structurer les exigences, un paragraphe par exigence, pas à pas, systématiquement, de la vue d'ensemble à la vue détaillée, basé sur une notion d'arbre d'exigences : exigences parents sous lesquelles se retrouvent des exigences enfants, les exigences parents étant la synthèse (vue d'ensemble) de leurs enfants ;
- Permet de lier chacune des exigences aux besoins du client ou exigences sources ;
- Offre une grande quantité d'options de formatage des documents générés ;
- Uniformise les documents d'exigences, parce que générés automatiquement ;
- Réduit la quantité d'erreurs et respecte des normes internationales.

## Réduction de la quantité d'erreurs

GenSpec réduit la quantité d'erreurs d'exigences :

- Génère un tableau de vérification : Sources Vs Exigences ;
- Facilite la couverture de l'ensemble des exigences : définition de tous les intrants et extrants sur les interfaces externes et liaison de chacun d'eux aux fonctions, et inversement ;
- Empêche d'introduire des incohérences de hiérarchie ou de liaison d'exigences ;
- Simplifie les corrections et mises à jour des exigences et des procédures d'évaluation, ces procédures étant définies avec les exigences ;
- Offre un vérificateur d'exigences (voir Figure 6) ;

– Permet d'utiliser des variables dans les textes d'exigences ;
– Gère un historique des modifications d'exigence ;
– Facilite la lecture et respecte des normes internationales.

## Respect des normes internationales

GenSpec respecte des normes internationales, dont les normes pertinentes de IEEE et de ISO/CEI. En effet, en plus des avantages ci-dessus, GenSpec :

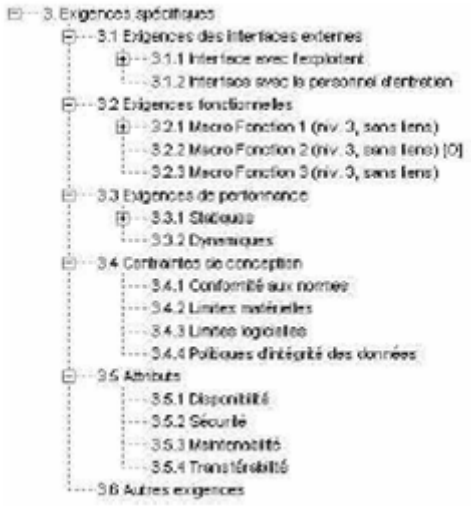
– Fixe un numéro de référence unique par exigence ;
– Oriente à énoncer des exigences validables, faisant abstraction des moyens de réalisation, les intrants et extrants d'exigences ne pouvant être que ceux des interfaces externes ;
– Facilite la modification des exigences, étant bien structurées et les liens entre exigences étant clairement définis : liens parent-enfant, liens intrant-extrant-fonction et autres liens (renvois) ;
– Définit la priorité de chacune des exigences.

# Cas d'utilisation

Cette dernière partie représente un cas d'utilisation de l'outil GenSpec sur un des projets en cours d'Horizon Vert, la gestion des plans agro-environnementaux de valorisation (PAEV).

## Création d'un projet

La première étape est la création du projet sur base d'un modèle contenu dans GenSpec. Dans le cadre de notre exemple, nous allons nous baser sur le premier modèle BD_Modele_IEEE_830 sans les interfaces. Il correspond principalement à la structure de la norme 830 de IEEE. Le modèle présenté dans GenSpec permet de rédiger un seul chapitre d'exigences, celui des exigences spécifiques. L'arbre de hiérarchie prédéfini de cette norme, à l'intérieur du modèle disponible dans GenSpec, représente bien cette définition :




Une fois le modèle choisit, il vous est demandé d'entrer le nom de votre projet et vos références telles que votre nom et mot de passe qui vous permettront d'ouvrir votre projet. Un avantage de GenSpec est qu'il travaille avec différents types d'autorisation : administrateur, utilisateur,... A chaque type d'autorisation correspond un certain nombre de fonctionnalités permises, l'administrateur ayant accès à toutes.

## Gestion des utilisateurs

Une fois entré dans le programme, vous pouvez ajouter, modifier ou supprimer un utilisateur via le menu “Session”.


## Manipulation des exigences

## Ajout d’une exigence

Dans la fenêtre **Arbre hiérarchique**, cliquer sur l’exigence parent de l’exigence à ajouter, avec le bouton de droite de la souris. Sélectionner l’option Ajouter sous cette exigence... Différents types d’exigences sont alors possibles :

- **Une contrainte de conception** désigne un élément susceptible de conditionner la réalisation des tâches, dans ce cas la conception en soi.
- **Une exigence de performance** est une capacité que doit posséder le système en termes de performance, une contrainte d’opération du système en disponibilité et en temps de réponse. Il s’agit d’un type d’exigence dite non fonctionnelle.
- **Une exigence fonctionnelle** est une fonctionnalité que doit rendre le système. Elle est à la base de la définition d’un système.

De plus, il est possible de définir plus précisément les exigences via les éléments suivants[18] :

- **Une macro fonction** est une fonction de très haut niveau, composée de fonctions.
- **Une fonction** est une tâche, action ou activité qui doit être exécutée pour parvenir à un résultat.
- **Un attribut** définit une caractéristique intrinsèque de l’élément sous lequel il est placé, tel que la sécurité, la performance. Ils ne peuvent être placés sous les exigences fonctionnelles.
- **Un détail** est une magnification d’une exigence, un niveau de définition supplémentaire. Il ne peut y en avoir plus de 7 par exigence.

## Méthodologie pour la construction de l’arbre hiérarchique

La première chose à savoir est que l’arbre possède 6 niveaux hiérarchiques. Afin de construire un arbre structuré et clair, il est impératif de travailler sur 5 ou 6 niveaux de hiérarchisation et de suivre certaines règles :

[18] Il est recommandé de ne pas utiliser les extrants et intrants

1. **Le niveau 1** concerne les exigences spécifiques.

2. **Le niveau 2** concerne les types d'exigences (fonctionnelles, performance,...).

3. **Le niveau 3** concerne les modules de vos exigences et n'est composé que de macro-fonctions :

   – Si votre programme doit gérer des utilisateurs, vous aurez en niveau 3 une macro-fonction "Gestion des utilisateurs"

   – Si votre programme doit gérer des boues, vous aurez en niveau 3 une marcro-fonction "Gestion des boues".

4. **Le niveau 4** peut être composé de deux éléments :

   – *Une macro-fonction* si vous jugez nécessaire de découper plus spécifiquement la macro-fonction parente en plusieurs macro-fonctions filles.

   – *Une fonction* qui représente une fonctionnalité spécifique de votre système futur (ajouter un utilisateur, ajouter une boue,...).

5. **Le niveau 5** peut aussi être composé de deux éléments :

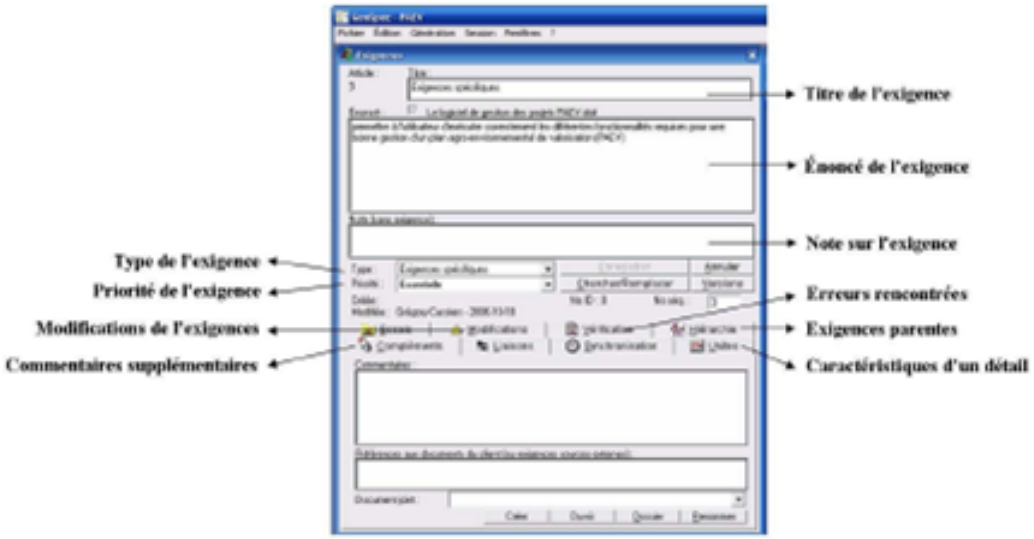
   – *Une fonction* si l'exigence parente est une macro-fonction. La fonction représente alors une fonctionnalité spécifique de votre système futur (ajouter un utilisateur, ajouter une boue,...).

   – *Un détail* si l'exigence parente est une fonction. Un détail est un élément nécessaire afin de réaliser la fonctionnalité (ex : le nom, le prénom, le mot de passe,... pour l'exigence "Ajout d'un utilisateur").

6. **Le niveau 6** ne peut contenir que des détails.

Certains éléments ne sont pas inclus dans cet arbre tels que les contraintes de conception ou les exigences de performances. Si celles-ci sont nécessaires, elles doivent être placées sur le niveau 2. Elles peuvent être ensuite découpées de la même manière que les exigences fonctionnelles : en contraintes de conception, en exigences de performances, en détail,...).

# Méthodologie pour la description des exigences




## Titre de l'exigence

Le titre contient le nom de l'exigence dans l'arbre. Pour chaque niveau, on pourra nommer les exigences de la façon suivante :

- **Niveau 1 :** Ce niveau ne possède qu'une seule exigence : les exigences spécifiques. Vous pouvez soit laisser ce titre inchangé ou soit opter pour un titre de la forme : "Exigences du projet PAEV". Il peut être considéré comme le titre de votre arbre.

- **Niveau 2 :** On retrouve à ce niveau les différents types d'exigences (fonctionnelles, performance,...). Ces titres ne doivent pas être changés.

- **Niveau 3 :** On retrouve à ce niveau des macro-fonctions. De façon générale, vous pouvez adopter un titre de la forme "Gestion de ..." tel que Gestion des boues,...

- **Niveau 4 :** Si c'est une macro-fonction, opter pour un titre de la forme "Gestion de ...". Si c'est une exigence, le titre doit correspondre au but de l'exigence, comme par exemple : ajouter un utilisateur, lister les boues,...

- **Niveau 5 :** Si c'est une exigence, le titre doit correspondre au but de l'exigence, comme par exemple : ajouter un utilisateur, lister les boues,... Si c'est un détail, le titre doit correspondre au nom du détail (nom, type,...).

- **Niveau 6 :** Ce niveau ne pouvant contenir que des détails, les titres doivent correspondre au nom des détails (nom, type,...).

### Énoncé de l'exigence

L'énoncé décrit ce que doit faire l'exigence, s'il est lié à un type d'exigence, une macro-fonction ou une fonction. De ce cas, il est impératif d'inclure dans votre texte d'énoncé qu'est-ce que l'exigence, comment la réaliser et qui est susceptible de la réaliser (quoi - comment - qui). Dans le cadre d'un détail, l'énoncé représente la description du détail.

*Exemple :* Ajouter un utilisateur

Le logiciel de gestion des PAEV doit permettre un accès sécurisé au système via une fenêtre d'authentification.

* **Quoi :** permettre un accès sécurisé au système.
* **Comment :** via une fenêtre d'authentification.
* **Qui :** l'utilisateur (celui-ci n'est pas présent directement dans l'énoncé mais sa présence est implicite suite à l'énoncé "Exigences du projet PAEV").

Dans l'énoncé, on peut aussi ajouter des contraintes liées à l'exigence. Pour un détail "date", on ajouterait alors la contrainte suivante : "Les dates doivent être choisies à l'aide d'une liste déroulante". Pour l'exigence "lister les champs", on ajouterait alors la contrainte suivante : "La liste de champs doit contenir la capacité de réception de phosphore et les besoins en N, P et K".

### Note sur l'exigence

L'utilisation des notes permet d'apporter des éléments supplémentaires qui ne sont pas relatifs aux exigences. Si vous ajoutez une note, il est préférable de réaliser un retour à la ligne dans l'énoncé afin d'avoir un document d'exportation clair. La note ne doit introduire aucune autre exigence. Une telle exigence serait dite cachée, et les développeurs risqueraient de ne pas en ternir compte en étudiant le document d'exigences

### Type de l'exigence

Le type de l'exigence se réfère au type que vous avez sélectionné lors de l'ajout de l'exigence dans l'arbre. Via la liste déroulante, il vous est possible de modifier le type d'une exigence sans devoir la supprimer et puis la rajouter.

### Priorité de l'exigence

Le degré de nécessité doit être choisi entre trois valeurs différentes, soit Essentielle, Complémentaire et Optionnelle. Le degré de nécessité d'une exigence enfant d'une exigence complémentaire peut très bien être essentielle. En effet, ce degré de nécessité est relatif à l'existence même du parent. Une exigence enfant est essentielle SI son exigence parent complémentaire ou optionnelle devient un événement vérifié.

### Modifications de l'exigence

Dans le cas où l'exigence est modifiée après sa création, l'onglet Modifications permet à l'utilisateur d'entrer un court texte spécifiant les différences entre la version antérieure et la nouvelle mise à jour. Comme il est possible avec le logiciel de revenir vers une version antérieure, les modifications correspondantes sont conservées dans le temps. Le texte entré dans l'onglet Modifications des exigences se retrouvera dans les documents générés "Listes des modifications courantes" et/ou dans "Historique des modifications".

### Vérification - Erreurs rencontrées

Sous l'onglet Vérification, GenSpec affiche les erreurs rencontrées lors de la vérification. Sous forme textuelle, toute erreur détectée à la suite du lancement de la commande vérification sera affichée sous cet onglet.

1. **Effectuer une vérification :** Pour effectuer une vérification, cliquer avec le bouton de droite de la souris sur l'exigence à valider dans l'arbre hiérarchique. Un menu contextuel apparaît. Cliquer sur l'option "Exploser avec vérification". Cela permet de développer tous les enfants de l'exigence mais en plus GenSpec validera les énoncés et structures de cette exigence sous certains points de vérification sélectionnables.

2. **Option de vérification :** Après avoir cliqué sur cette option, une fenêtre apparaît et offre plusieurs options de vérification. Les règles qui ne seront pas respectées lors de la vérification d'une exigence seront affichées dans la rubrique Erreurs de l'onglet Vérification[19].

### Hiérarchie - Exigences parentes

Dans cette fenêtre, vous pouvez consulter les exigences parents de l'exigence sélectionnée dans l'arbre hiérarchique de droite. Via cette fenêtre, il n'est pas possible de modifier les exigences parentes, son seul but est la consultation.

### Compléments - Commentaires supplémentaires

Dans le cas où des détails sur l'énoncé doivent être fournis, sur les différents comportements remarqués ou tout autre point sur lequel porter son attention lors du développement, c'est l'espace à utiliser. Lorsqu'un commentaire est entré dans la fiche d'une exigence, le titre de cette exigence devient vert dans l'arbre hiérarchique. Il est possible lors de la génération de documents Word d'inclure ou non ces commentaires.

[19]L'option de la rubrique "Vérification des liens intrant-extrant-fonction ne doivent pas être cochées puisque les intrants et extrants ne sont pas utilisés dans le projet.

### Unités - Caractéristiques d'un détail

Les unités d'une mesure sont à spécifier pour s'assurer que les différents partis se basent sur à la même échelle. L'entrée de valeurs dans les champs de l'onglet Unités ne s'applique uniquement qu'aux détails[20]. Sous cet onglet, on retrouve cinq paramètres ajustables, reliés aux unités :

- **Unités :** L'unité proprement dite du détail (ex : tonnes pour le tonnage restant de l'exigence "Sélectionner les sites".
- **Plage :** La plage des valeurs possibles et permises (ex : un login compris entre 100000 et 999999 lors de l'identification d'un utilisateur).
- **Valeur par défaut :** La valeur par défaut s'il en existe une (ex : la valeur par défaut utilisateur pour le statut lors d'un ajout d'utilisateur).
- **Précision :** Degré de précision de la valeur entrée (ex : Précision de 1 pour le tonnage lors d'un ajout d'un site c'est-à-dire une précision à la tonne).

## Déplacement d'une exigence

Dans l'arbre hiérarchique, après avoir cliqué sur l'exigence à déplacer, cliquer avec le bouton de droite de la souris sur cette même sélection. Un menu contextuel apparaît. Ce second bloc d'options du menu contextuel de l'arbre hiérarchique permet de déplacer l'exigence sélectionnée au Niveau suivant ou au Niveau Précédent.

## Génération de document

La dernière fonctionnalité que je vais vous présenter sur GenSpec est sa génération de document. Via le menu "Génération" → "Générer les documents", vous obtenez la fenêtre suivante :


Il vous suffit alors de sélectionner le document que vous voulez générer et d'appuyer sur "Générer les documents". Votre document est alors généré sur base du modèle contenu dans le répertoire du projet.

[20] Intrants et extrants n'étant pas utilisés

## Options de génération

Dans l'arbre hiérarchique, l'onglet Options de génération regroupe toutes les options relatives à la génération de documents. Voici une courte description de l'effet des options principales sur les documents générés :

- **Énoncé :** Permet d'inclure dans les documents générés l'énoncé des exigences.
- **Unités :** Cliquer sur cette option pour afficher l'unité des exigences, sa valeur par défaut, sa précision, sa résolution et sa plage.
- **Degré de nécessité :** Inclus dans les documents le degré de nécessité (essentielle, complémentaire ou optionnelle) des exigences.
- **Document joint :** Permet à l'utilisateur d'inclure les schémas et figures joints à partir d'autres fichiers spécifiés.
- **Titre :** Inclus le titre des exigences.
- **Saut de paragraphe avant l'énoncé :** Effectue, après le titre des exigences, un saut de paragraphe pour améliorer la lecture des documents.
- **Note :** Permet d'inclure les notes sans exigences dans les documents générés.

De plus, le nombre de niveaux possibles de l'Arbre hiérarchique généré dans les documents est spécifiable par un menu déroulant, qui permet une sélection entre 1 et 6.

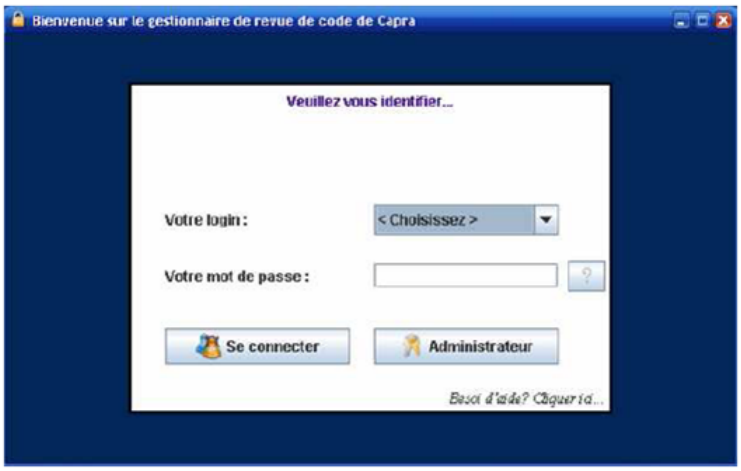
# Rapport H :

*CodeReview - Manuel d'aide*

# Ecran de démarrage

## S'identifier

Lorsque vous démarrez le gestionnaire de revue de code, la première fenêtre qui s'affiche est semblable à celle ci-dessous :




La première étape avant d'utiliser le gestionnaire de revue de code est de vous identifier auprès de celui-ci grâce au mot de passe et au login (6 chiffres) que vous avez reçus par e-mail lors de la création de votre compte. Pour cela, vous devez sélectionner dans la liste déroulante votre login et ensuite, taper votre mot de passe. Deux solutions s'offrent alors à vous :

**Vous connecter en administrateur :** Ce type de connexion n'est possible que si votre compte possède des droits d'administrateur. Il vous offre des fonctionnalités relatives à la gestion des utilisateurs.

**Vous connecter en utilisateur :** Ce type de connexion est possible pour toutes personnes enregistrées dans le système. Il vous offre des fonctionnalités relatives aux révisions de code et aux références (vos fichiers de code).

## Récupérer son mot de passe

La récupération de votre mot de passe peut se réaliser grâce au bouton “?” qui se situe à droite du champ de saisie de votre mot de passe. En cliquant dessus, vous obtenez le message suivant :

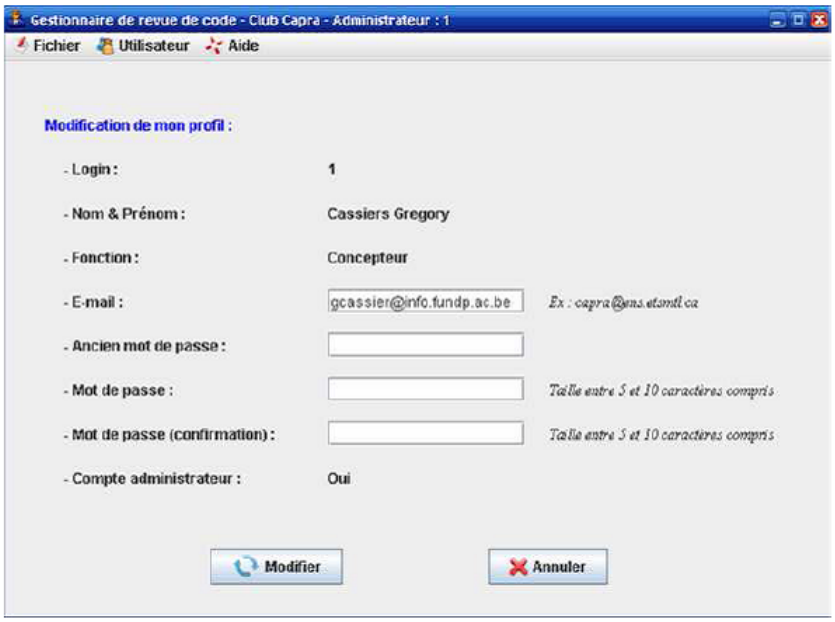

Si vous cliquer sur “oui”, un e-mail sera envoyé sur votre boite de courriel électronique. Celui-ci contiendra les caractéristiques de votre profil telles que reçues lors de la création de votre compte.

## Besoin d’aide ?

Si vous avez besoin d’aide à partir de cette fenêtre d’identification, il vous suffit de cliquer sur le lien “Besoin d’aide ? Cliquer ici...”. Une fois dans le gestionnaire, vous pouvez accéder à l’aide par la touche “F1” ou par le menu “Aide”.


# Menu “Fichier”

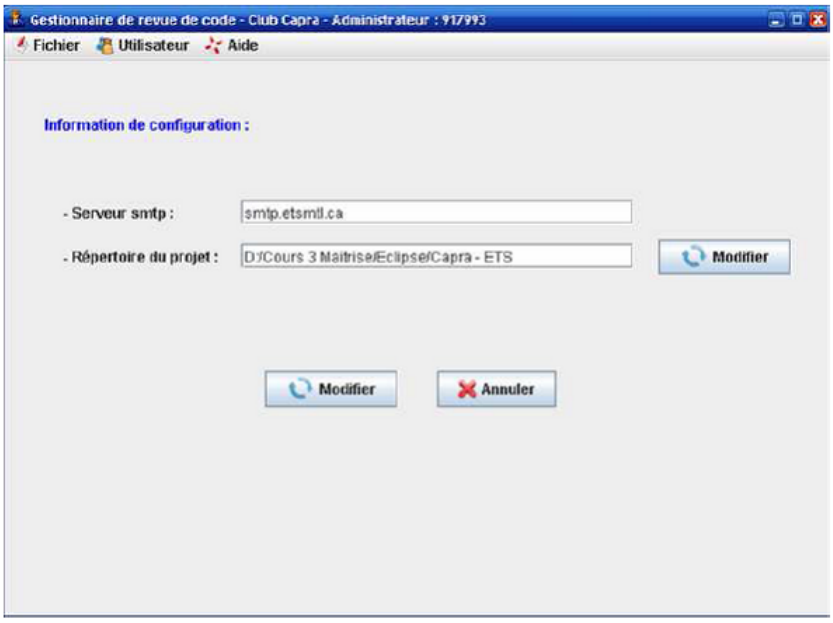
## Modifier son profil



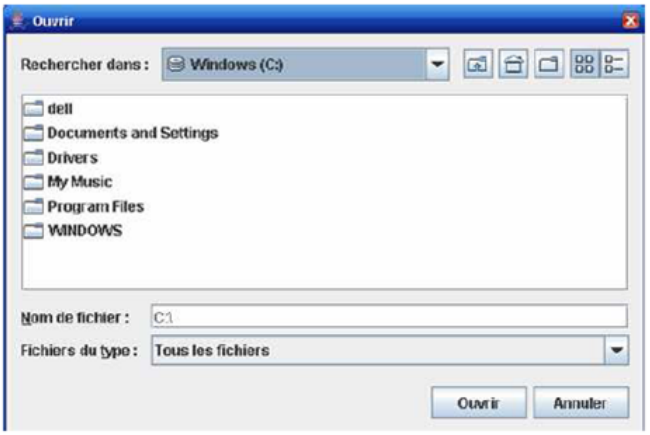

Cette fonctionnalité vous permet de modifier certaines caractéristiques de votre compte administrateur, à savoir :

- **Votre e-mail :** L'e-mail doit être une adresse valide (ex : capra@ens.etsmtl.ca). Elle vous permet de recevoir les modifications effectuées sur votre compte.
- **Votre mot de passe :** Pour modifier votre mot de passe, vous devez d'abord entrer votre ancien mot de passe et ensuite, entrer votre nouveau mot de passe ainsi qu'une confirmation de celui-ci afin de vérifier que le mot de passe entré est bien celui que vous désirez. La longueur de votre nouveau mot de passe doit être comprise entre 4 et 10 caractères inclus.

## Configuration

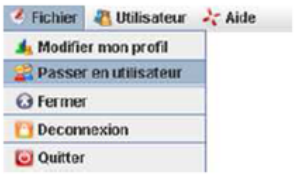



Via cette fonctionnalité, il vous est possible de modifier le serveur SMTP nécessaire à la gestion des e-mails ainsi que le chemin relatif au répertoire de votre projet. Pour modifier ce chemin, cliquer sur le bouton “Modifier”, une fenêtre comme ci-dessous apparaît alors et vous permet de sélectionner le répertoire relatif à votre projet.



## Passer en utilisateur

Passer en utilisateur vous permet de passer directement à l'interface de l'utilisateur sans devoir vous déconnecter du système et vous reconnecter en mode utilisateur.




## Déconnexion

En cliquant sur "Déconnexion", vous revenez à l'interface de démarrage du gestionnaire de revue de code.


## Quitter

En cliquant sur "Quitter", vous fermez le gestionnaire de revue de code.

# Menu “Utilisateur”

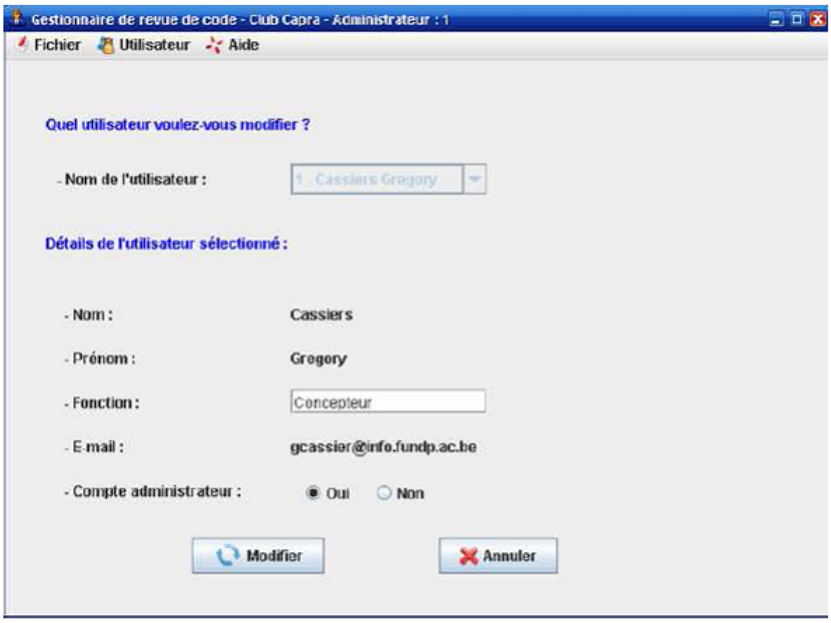
## Ajouter un utilisateur


La première fonctionnalité relative aux utilisateurs concerne l'ajout d'un nouvel utilisateur. Dans cette optique, vous devez remplir les champs suivants :

- **Nom :** Le nom du nouvel utilisateur.
- **Prénom :** Le prénom du nouvel utilisateur.
- **Fonction :** La fonction du nouvel utilisateur au sein du Club Capra.
- **E-mail :** L'adresse e-mail de l'utilisateur. Elle doit être valide (ex : capra@ens.etsmtl.ca). Elle permet au nouvel utilisateur de recevoir les caractéristiques de son compte sur sa boîte e-mail.
- **Mot de passe :** Le mot de passe de l'utilisateur. La longueur du mot de passe doit être comprise entre 4 et 10 caractères inclus. Une confirmation de celui-ci vous est demandée afin de vérifier que le mot de passe entré est celui que vous désirez.
- **Compte administrateur :**
  - *Oui :* l'utilisateur pourra se connecter en tant qu'administrateur et influencer sur la gestion des utilisateurs.
  - *Non :* l'utilisateur ne possèdera pas de droit d'administrateur sur son compte. C'est le statut par défaut.

## Modifier un utilisateur

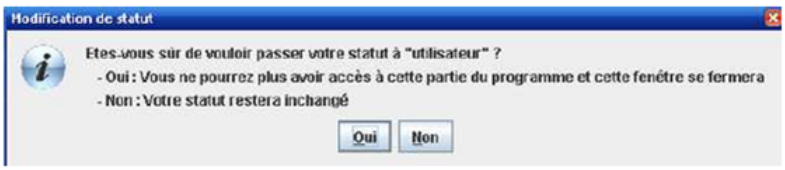
Avant de modifier le compte d'un utilisateur, vous devez d'abord le sélectionner dans la liste déroulante et appuyer sur le bouton "Détails".




Une fois l'utilisateur sélectionné, vous pouvez alors modifier certaines de ses caractéristiques, à savoir :

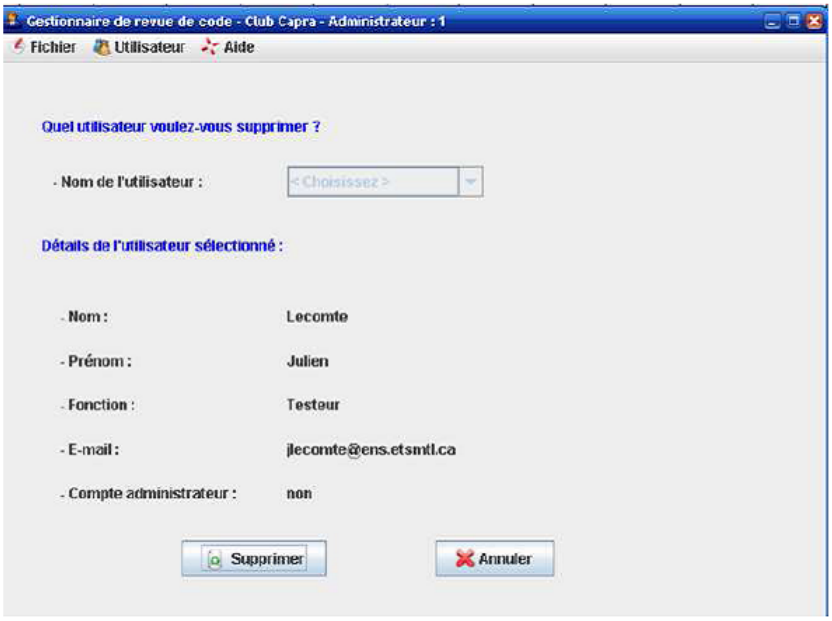
- **Fonction :** La fonction de l'utilisateur au sein du Club Capra.
- **Compte administrateur :**
  - *Oui :* l'utilisateur pourra se connecter en tant qu'administrateur et influencer sur la gestion des utilisateurs.
  - *Non :* l'utilisateur ne possèdera pas de droit d'administrateur sur son compte. C'est le statut par défaut.

Cette fonctionnalité peut aussi s'appliquer sur votre compte. C'est pourquoi, si vous modifier votre statut à celui d' "Utilisateur" en cochant l'option "Non" de la caractéristique "Compte administrateur", un popup (voir ci-dessous) s'affichera et vous demandera de confirmer votre choix. En cas de confirmation, vous passerez directement en mode utilisateur et vous ne pourrez plus accéder à cette partie du gestionnaire de revue de code. Dans le cas contraire, seul la modification de votre fonction sera effectuée à votre compte.



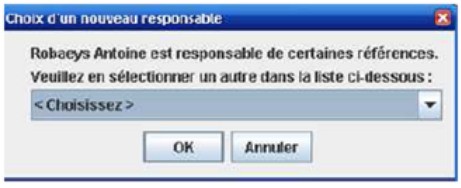
## Supprimer un utilisateur

Cette troisième fonctionnalité vous permet de supprimer un utilisateur si celui-ci ne faisait par exemple plus partie du Club. Pour cela, vous devez sélectionner l'utilisateur dans la liste déroulante et ensuite, appuyer sur le bouton "Détails".

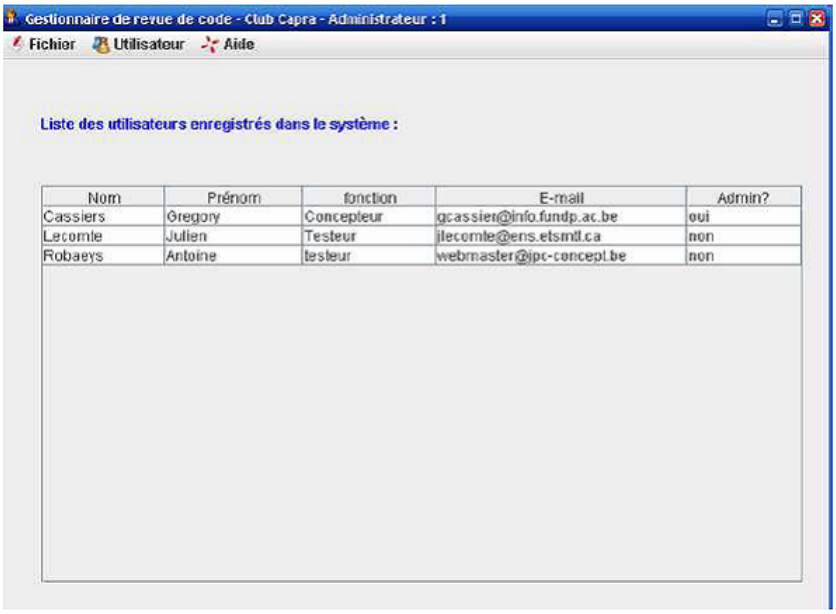



Vous obtenez alors les caractéristiques du compte utilisateur que vous avez sélectionné. Il vous est ainsi possible de vérifier que l'utilisateur est bien celui recherché et de le supprimer.

Si vous supprimer un utilisateur qui est responsable de certaines références enregistrées dans le système, un popup (voir ci-dessous) s'affiche et vous demande de sélectionner un nouveau responsable pour ces références.



## Lister les utilisateurs



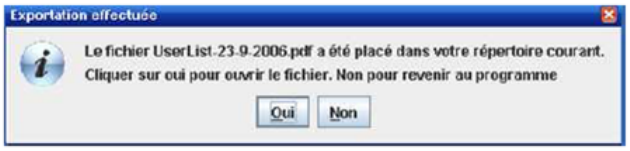

Cette avant dernière fonctionnalité vous permet de lister l'ensemble des utilisateurs et leurs caractéristiques, à savoir : le nom, le prénom, la fonction, l'adresse e-mail et le statut (administrateur = oui, utilisateur = non)[21].

## Exporter les utilisateurs


La dernière fonctionnalité applicable aux utilisateurs concerne l'exportation de ceux-ci. En effet, vous pouvez exporter les utilisateurs et leurs caractéristiques au format pdf. Pour cela,

[21] Les tailles des colonnes sont modifiables manuellement si une des données est entrecoupée

sélectionner les caractéristiques que vous voulez inclure dans le fichier d'exportation et modifier le nom du fichier d'exportation si vous le désirez (sans extension).

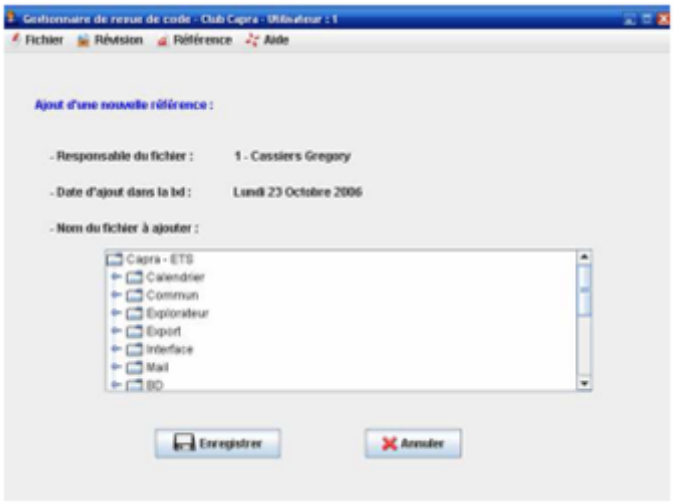



Une fois le fichier d'exportation généré, un popup s'affiche et vous propose d'ouvrir le fichier. Cliquer sur oui pour l'ouvrir, sur non pour revenir dans le gestionnaire de revue de code[22].

[22]Les fichiers d'exportation sont situés dans le répertoire d'exécution du jar du gestionnaire de revue de code

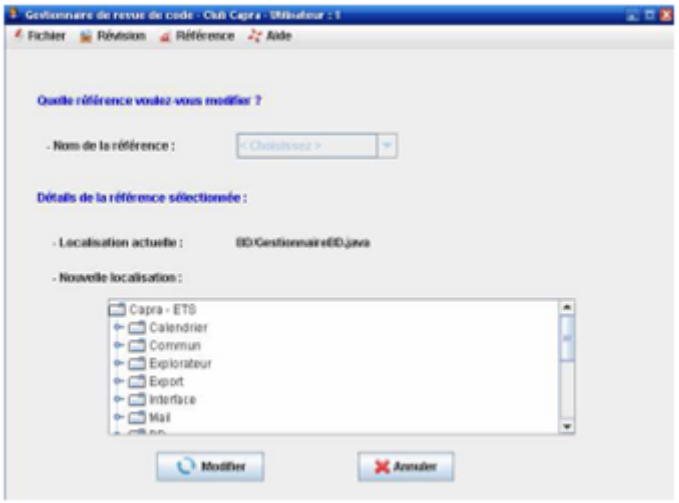
# Menu “Référence”

## Ajouter une référence




La première fonctionnalité applicable sur les références concerne l'ajout de celles-ci. Pour cela, sélectionner la référence dans l'arbre hiérarchique et appuyer sur le bouton “Enregistrer”.

## Modifier une référence



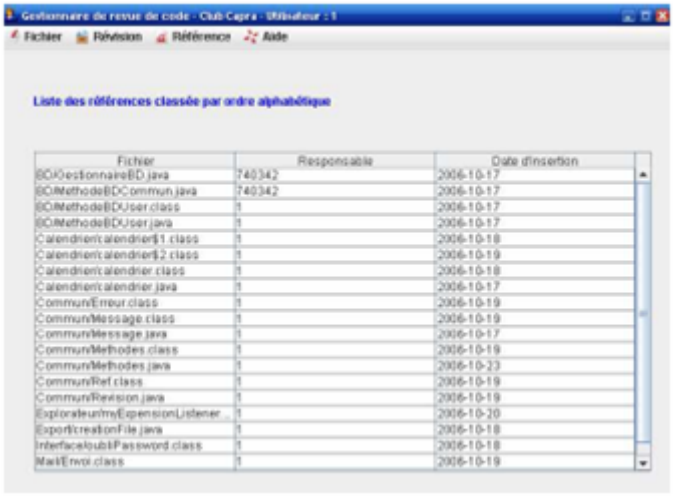

Pour modifier une référence, sélectionner d'abord la référence dans la liste déroulante et appuyer sur le bouton “Détails”. L'ancienne localisation de la référence s'affiche alors et vous

pouvez la modifier en sélectionnant la nouvelle localisation dans l'arbre hiérarchique. Il ne vous reste plus qu'à valider en appuyant sur "Modifier".

## Supprimer une référence


Pour supprimer une référence, sélectionner d'abord la référence dans la liste déroulante et appuyer sur le bouton "Détails". Il ne vous reste alors plus qu'à valider la suppression en appuyant sur "Supprimer".

## Lister les références




Cette avant dernière fonctionnalité vous permet de lister l'ensemble des références et leurs caractéristiques, à savoir : le nom de la référence, son responsable et sa date d'insertion dans le système.

## Exporter les références

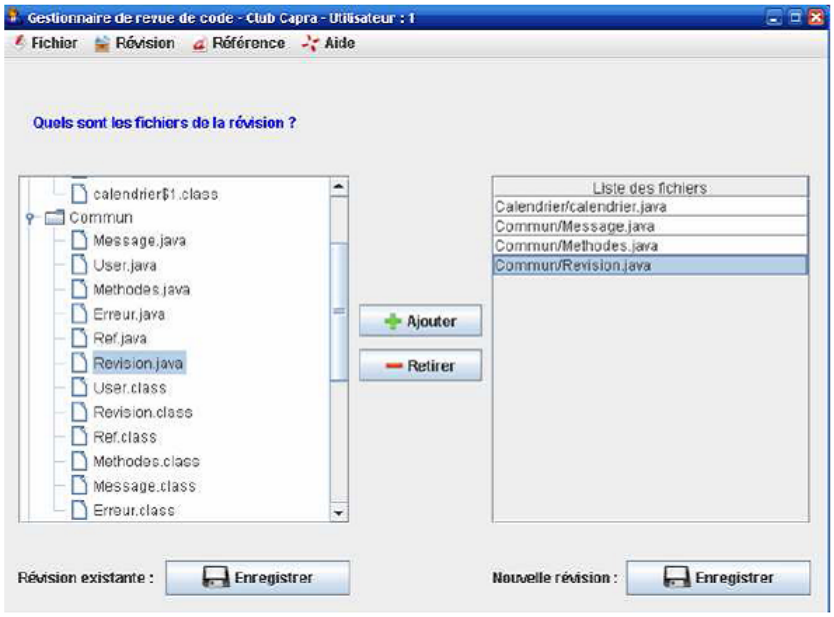

La dernière fonctionnalité applicable aux références concerne l'exportation de celles-ci. En effet, vous pouvez exporter les références et leurs caractéristiques au format pdf. Pour cela, sélectionner les caractéristiques que vous voulez inclure dans le fichier d'exportation et modifier le nom du fichier d'exportation si vous le désirez (sans extension).


Une fois le fichier d'exportation généré, un popup s'affiche et vous propose d'ouvrir le fichier. Cliquer sur oui pour l'ouvrir, sur non pour revenir dans le gestionnaire de revue de code[23].

[23]Les fichiers d'exportation sont situés dans le répertoire d'exécution du jar du gestionnaire de revue de code

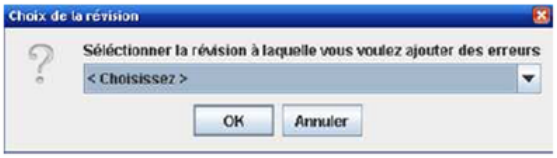
# Utilisateur - Menu “Révision”

## Ajouter une révision




Pour ajouter une nouvelle révision, vous devez d’abord sélectionner les fichiers de la révision dans l’arbre hiérarchique de gauche et les ajouter via le bouton “ajouter”. Les fichiers sont alors insérés dans la liste de droite. Vous pouvez supprimer un fichier de cette liste via le bouton “Retirer”. Une fois les fichiers sélectionnés, deux possibilités s’offrent à vous :

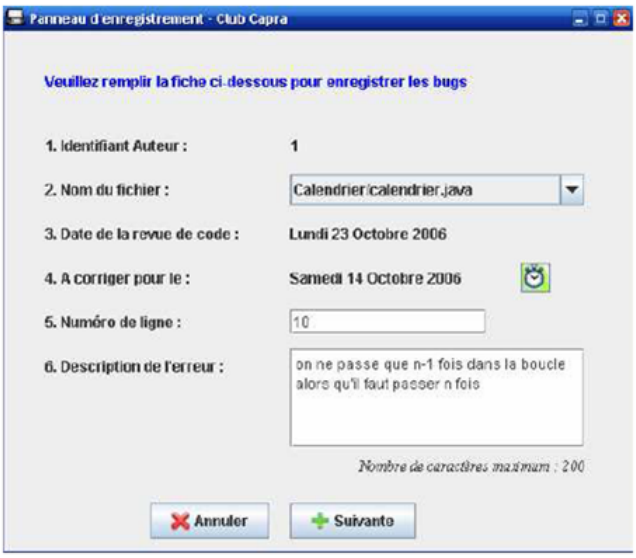
– **Révision existante :** Choisissez cette option si vous voulez insérer de nouvelles erreurs dans une révision que vous avez déjà faite. La sélection de cette révision existante se fait par le popup suivant :




– **Nouvelle révision :** Choisissez cette option si vous désirez ajouter une nouvelle révision dans le système.

Si certaines références inclues dans votre révision non pas encore été enregistrées dans le système, le gestionnaire vous propose de les enregistrer en vous nommant responsable de celles-ci. Si vous refusez de les enregistrer, ces références ne seront pas inclues dans votre révision et vous ne pourrez donc pas enregistrer les erreurs s’y rapportant.


Il ne vous reste alors plus qu'à enregistrer les erreurs de votre révision. Pour cela, une nouvelle fenêtre composée des éléments suivants s'ouvre :




– **Nom du fichier :** Sélectionner dans la liste déroulante le nom du fichier pour lequel l'erreur doit être enregistrée. La liste est composée des fichiers que vous avez sélectionnés préalablement.

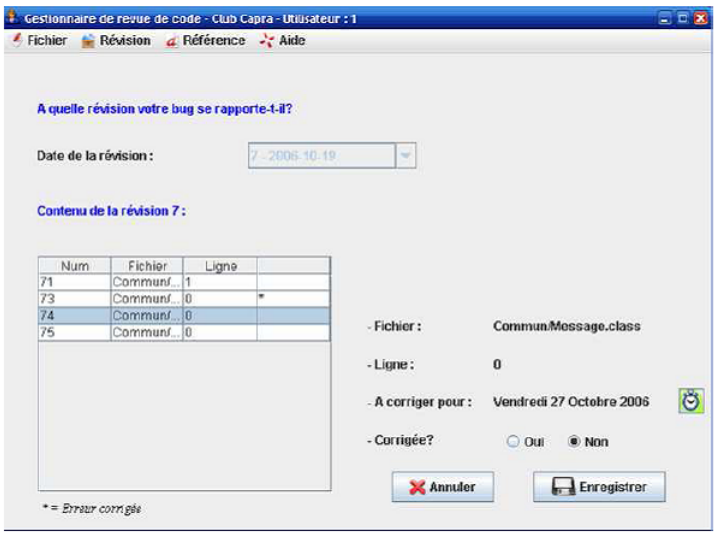
– **A corriger pour le :** Sélectionner, via le bouton, la date pour laquelle l'erreur doit être corrigée. Ce bouton vous ouvre un calendrier où vous devez sélectionner la date et appuyer sur "Appliquer" pour la valider.

– **Numéro de ligne :** Ce champ doit contenir le numéro de ligne où l'erreur a été détectée dans le fichier.
– **Description de l'erreur :** Ce champ doit contenir la description de l'erreur. Sa taille est limité à 200 caractères.

Une fois les différents champs remplis, plusieurs possibilités s'offrent à vous via les boutons suivants :

– **Suivante :** Cliquez sur ce bouton si vous désirez ajouter une nouvelle erreur dans votre révision. Les erreurs ne sont alors pas encore enregistrées dans le système.
– **Enregistrer :** Cliquez sur ce bouton si vous désirez terminer l'ajout de nouvelles erreurs pour cette révision et les enregistrer dans le système. Si certains fichiers inclus dans votre révision ne contiennent pas d'erreur, ceux-ci sont quand même ajoutés à la révision avec comme description "aucune erreur".
– **Annuler :** Cliquez sur ce bouton afin de revenir à la fenêtre précédente et ainsi, annuler l'enregistrement des erreurs en cours.

## Modifier une révision

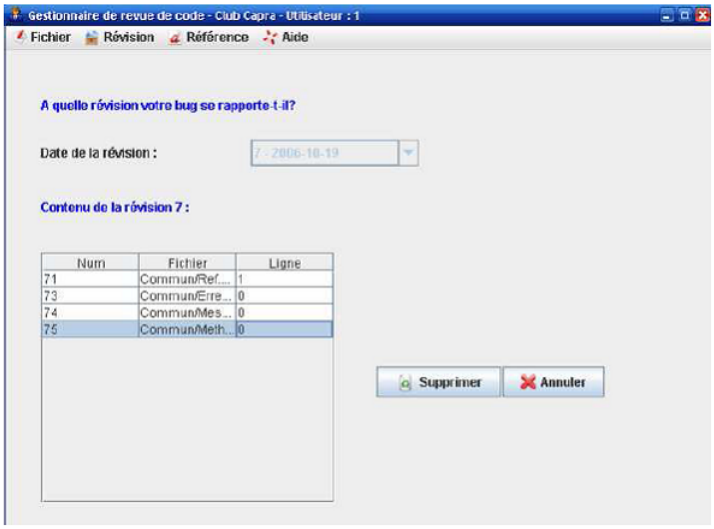



Cette fonctionnalité vous permet de modifier certaines caractéristiques des erreurs contenues dans les révisions. Pour cela, sélectionner la révision dans la liste déroulante et appuyer sur "Détails". La liste des erreurs de la révision sélectionnée s'affiche à l'écran. Vous n'avez plus qu'à sélectionner l'erreur que vous voulez modifier dans cette liste et appuyer sur le bouton "Modifier". Deux caractéristiques peuvent alors être modifiées :

– **A corriger pour le :** Ce champ contient la date prévue pour la correction. Vous pouvez la modifier via le bouton situé à la droite de la date. Ce bouton vous ouvre un calendrier où vous devez sélectionner la date et appuyer sur "Appliquer" pour la valider.

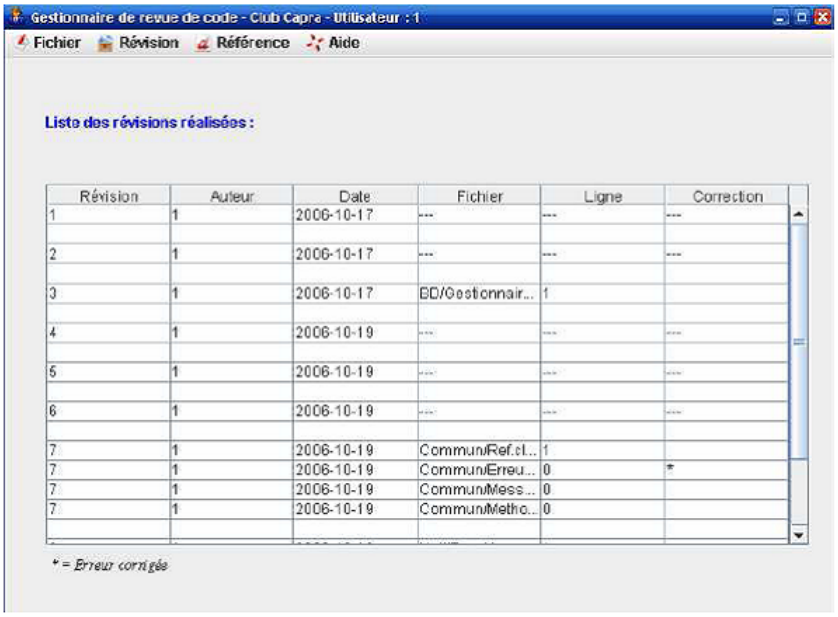

- **Corrigée ? :**
  - *Oui :* Signifie que l'erreur a été corrigée. La date "A corriger pour" se change automatiquement à celle d'aujourd'hui.
  - *Non :* Signifie que l'erreur n'a pas encore été corrigée et qu'elle doit être corrigée pour la date spécifiée dans "A corriger pour".

## Supprimer une révision



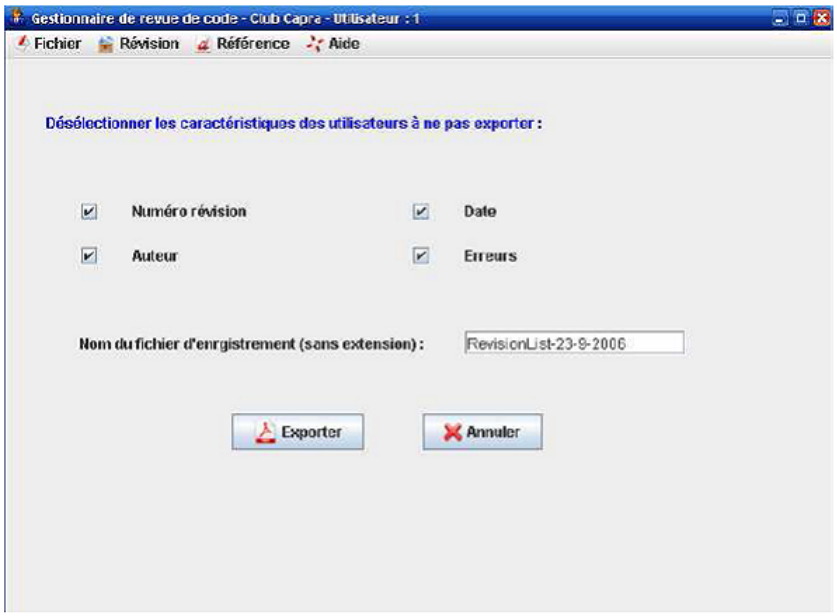

Cette fonctionnalité vous offre la possibilité de supprimer des erreurs contenues dans une révision. Pour cela, vous devez sélectionner la révision dans la liste déroulante et appuyer sur le bouton "Détails". Les erreurs de la révision sélectionnée s'affichent alors sur l'interface et vous n'avez plus qu'à sélectionner l'erreur dans la liste et la supprimer.

## Lister les révisions




Cette avant dernière fonctionnalité vous permet de lister l'ensemble des révisions et des erreurs de celles-ci et des caractéristiques de ces erreurs, à savoir : le numéro de la révision, l'auteur de la révision, la date de la réalisation de la révision, le fichier qui contient l'erreur, la ligne où se situe l'erreur et si cette erreur a déjà été corrigée ou non.

## Exporter les révisions




La dernière fonctionnalité applicable aux révisions concerne l'exportation de celles-ci. En effet, vous pouvez exporter les révisions et leur contenu au format pdf. Pour cela, sélectionner

les caractéristiques que vous voulez inclure dans le fichier d'exportation et modifier le nom du fichier d'exportation si vous le désirez (sans extension).


Une fois le fichier d'exportation généré, un popup s'affiche et vous propose d'ouvrir le fichier. Cliquer sur oui pour l'ouvrir, sur non pour revenir dans le gestionnaire de revue de code[24].

[24]Les fichiers d'exportation sont situés dans le répertoire d'exécution du jar du gestionnaire de revue de code

# Rapport I :

*Modex - Manuel d'aide*

# Gestion de la base de données

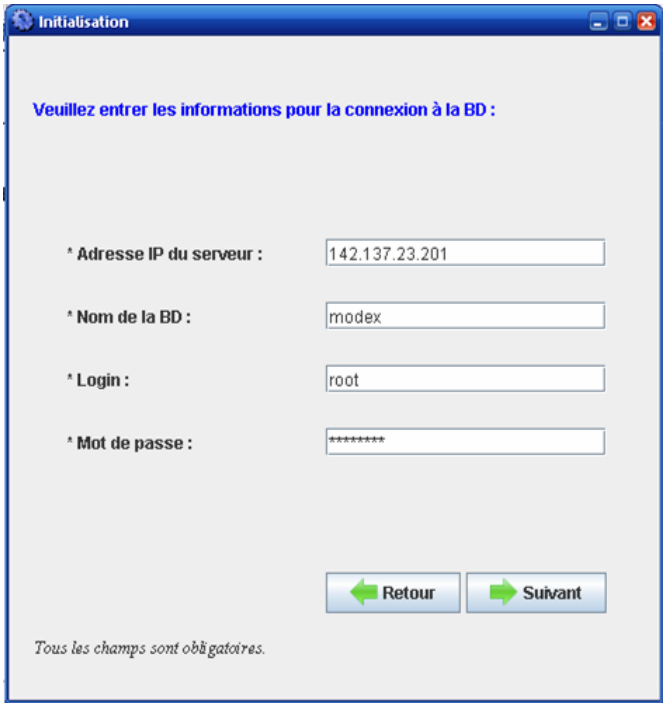
## Initialisation de modeX

ModeX utilisant une base de données, la phase d'initialisation permet, lors du premier démarrage de modeX sur un poste de travail, soit, d'initialiser une base de données, soit, de se connecter à une base de données existante.

ModeX détectera automatiquement si la base de données spécifiée ci-après est déjà initialisée sur le serveur. Le cas échéant, il permettra à l'utilisateur de connecter modeX à cette base de données. Dans le cas contraire, il permettra au chef de projet de créer les tables de la base de données de modeX.

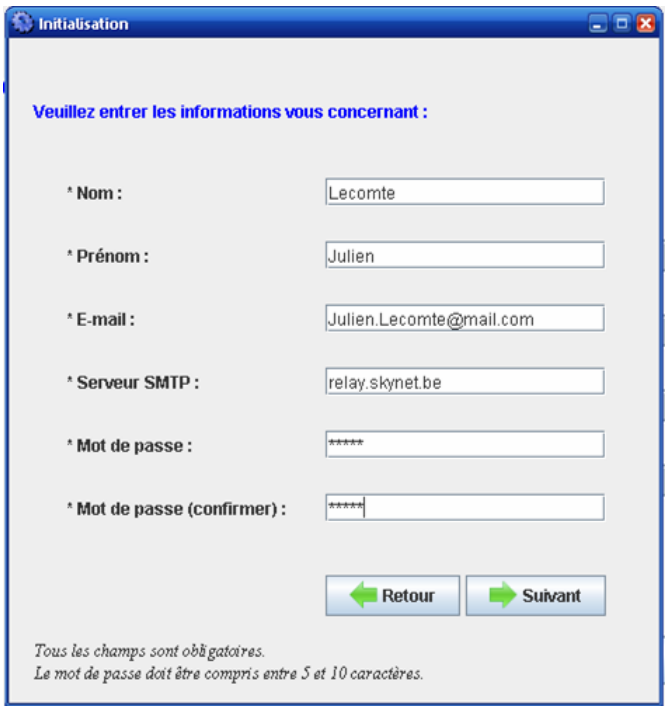

## Gestion de la base de données

Afin de pouvoir se connecter à la base de données, il est nécessaire de spécifier l'adresse IP du serveur de base de données de l'entreprise. Les champs en dessous de l'adresse IP - nom de la BD, login et mot de passe - permettent de spécifier la base de données à laquelle modeX doit se connecter.




## Gestion de votre compte

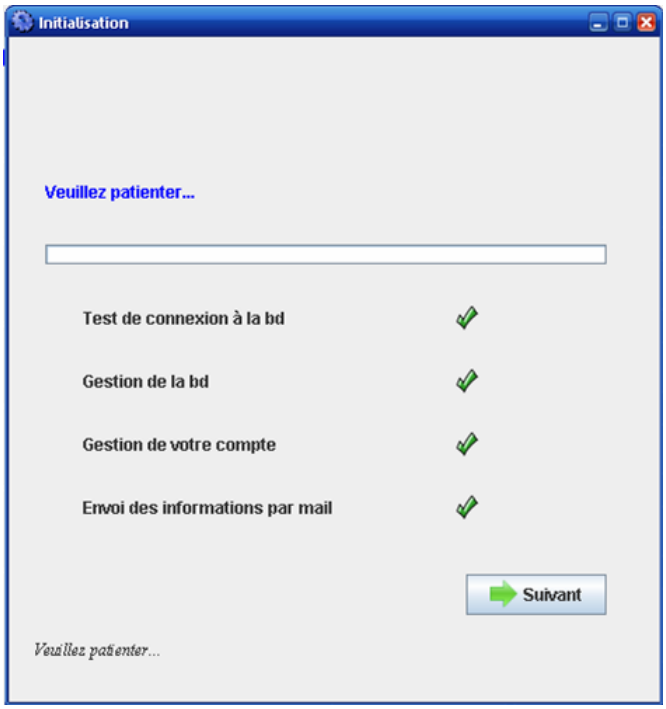
Les informations à introduire à cette étape concernent soit, l'utilisateur potentiel du programme modeX, soit, le chef de projet qui initialise la base de données. Dans le premier cas, ces informations seront envoyées par mail au chef de projet qui donnera son approbation en ajoutant, ultérieurement, l'utilisateur dans le système. Dans le second cas, ces informations permettront la création du compte « Chef de projet » dans le système.



Remarquons que le serveur smtp étant généralement le serveur de l'entreprise, il est toutefois possible de le personnaliser pour une éventuelle utilisation hors des locaux de l'entreprise.

## Installation

L'étape d'installation permet de :
- Tester la connexion à la base de données.
- Détecter la base de données existante ou créer les tables de la nouvelle base de données.
- Ajouter éventuellement le compte « Chef de projet » au système.
- Envoyer par mail les informations de l'utilisateur au chef de projet ou envoyer les informations du compte « Chef de projet ».




## Fin

La phase d'initialisation se termine en récapitulant les informations. En cliquant sur terminer, le programme se lancera directement et permettra l'identification de l'utilisateur ou du chef de projet. Remarquons que l'utilisateur ne pourra se connecter à modeX que lorsque le chef de projet l'aura explicitement accepté et introduit dans le système.

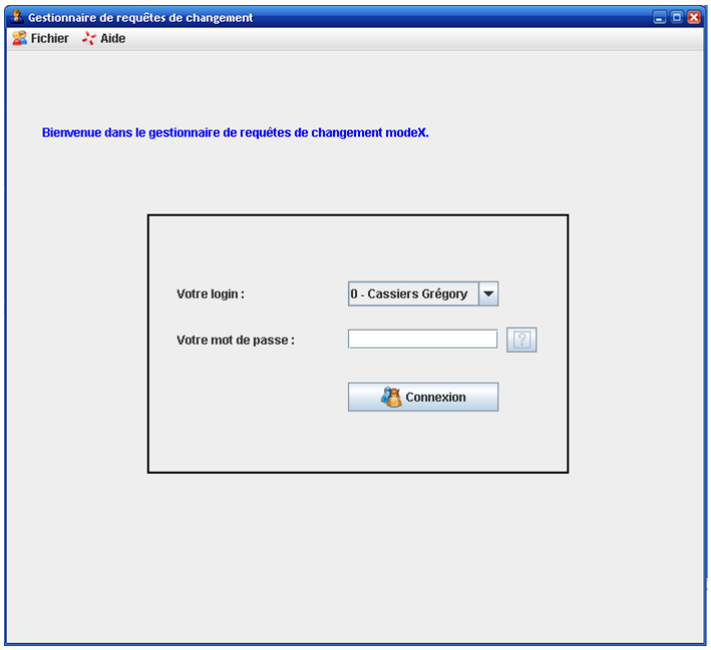
# Identification

## S'identifier

Lors du démarrage de modeX, la première fenêtre qui s'affiche permet de s'identifier dans le système. Il vous faudra donc posséder un compte dans ce dernier, à savoir, un login et un mot de passe.

Le login est composé d'un numéro d'identification unique, généré automatiquement lors de la création du compte, ainsi que du nom et prénom de la personne. Le mot de passe doit, quant à lui, être composé de minimum 5 et maximum 10 caractères.

Suivant votre type de compte - « Chef de projet » ou « Utilisateur » - vous pourrez accéder ou non à certaines fonctionnalités de modeX. Nous verrons ci-après les fonctionnalités disponibles pour chacun des deux types de compte.




## Récupérer son mot de passe

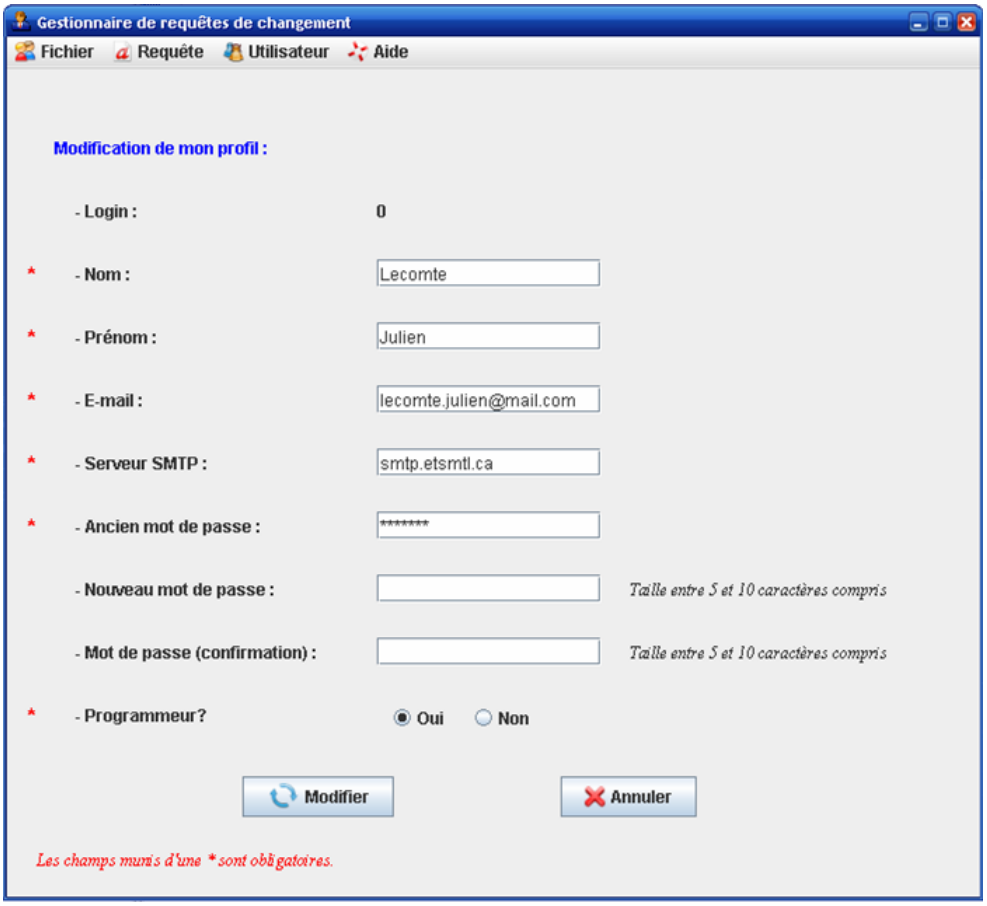
La récupération de votre mot de passe peut se réaliser via le bouton « ? » qui se situe à droite du champ de saisie du mot de passe. En cliquant dessus, votre mot de passe sera envoyé à l'adresse mail que vous avez spécifié dans votre compte.

# Menu "Fichier"

## Modifier le profil

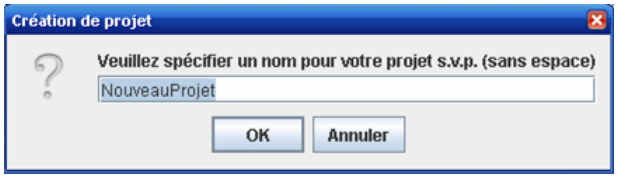
Cette fonctionnalité permet au chef de projet et à l'utilisateur de modifier les caractéristiques de leur compte.

La modification de ces informations nécessite la réintroduction du mot de passe courant du compte. Il est également possible de modifier ce mot de passe en spécifiant dans les deux derniers champs le nouveau mot de passe désiré.



## Nouveau projet

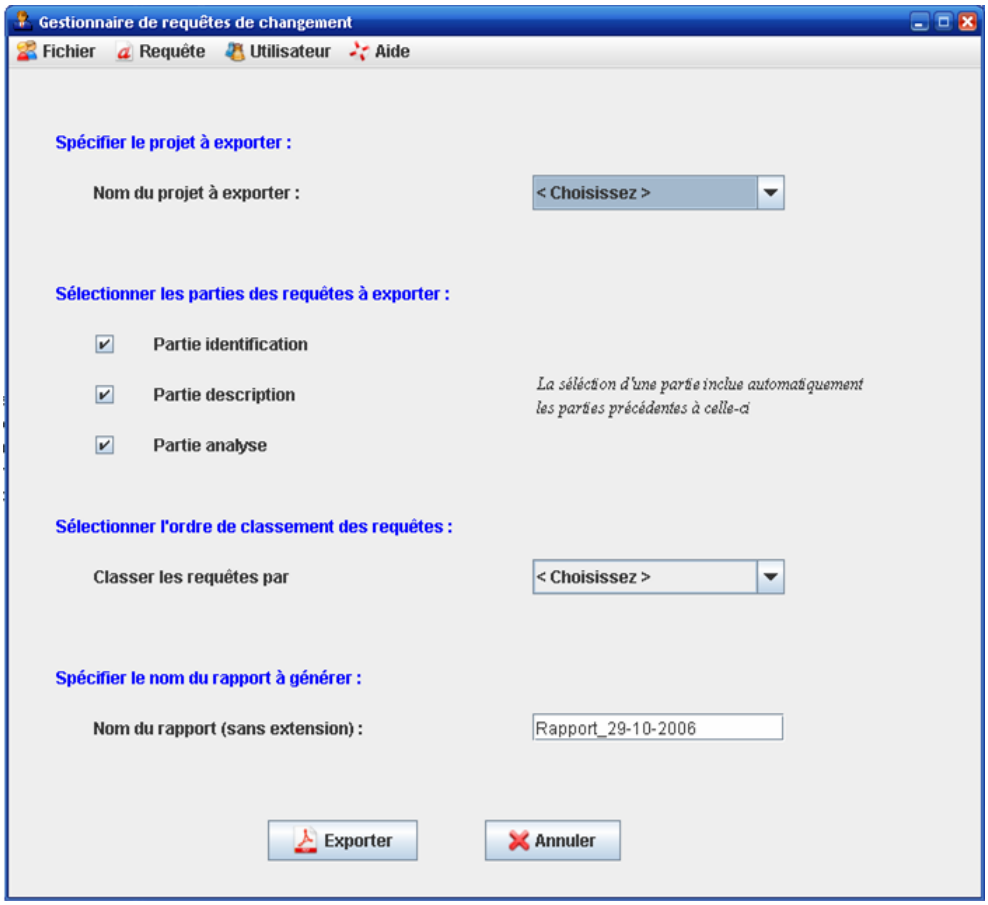
Le chef de projet peut créer des projets afin de pouvoir gérer les requêtes de changement pour chacun de ces projets en particulier. Ceci entraîne la création d'un dossier avec le nom du projet dans le répertoire courant de l'application.




## Exporter le projet

Le chef de projet et l'utilisateur peuvent générer au format pdf un rapport contenant l'ensemble des requêtes d'un projet particulier. Ce document sera placé dans un dossier « Rapport » situé dans le dossier du projet.

Pour chaque requête, il est permis de choisir les parties d'information que l'on veut générer telles que l'identification, la description et l'analyse.

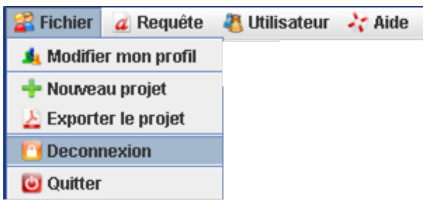


Remarquons que, dans un but de clarté, le choix d'une partie implique automatiquement le choix des parties précédentes. Par exemple, le choix de la partie description, inclura également la partie identification.

En outre, il est possible d'organiser les requêtes par critère. Par exemple, on souhaitera voir apparaître les requêtes d'un projet par ordre de priorité.

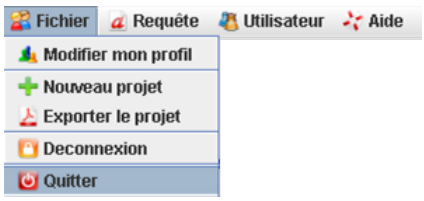
## Déconnexion

En cliquant sur déconnexion, vous revenez à l'interface d'identification de modeX.




## Quitter

En cliquant sur quitter, vous fermez le programme modeX.



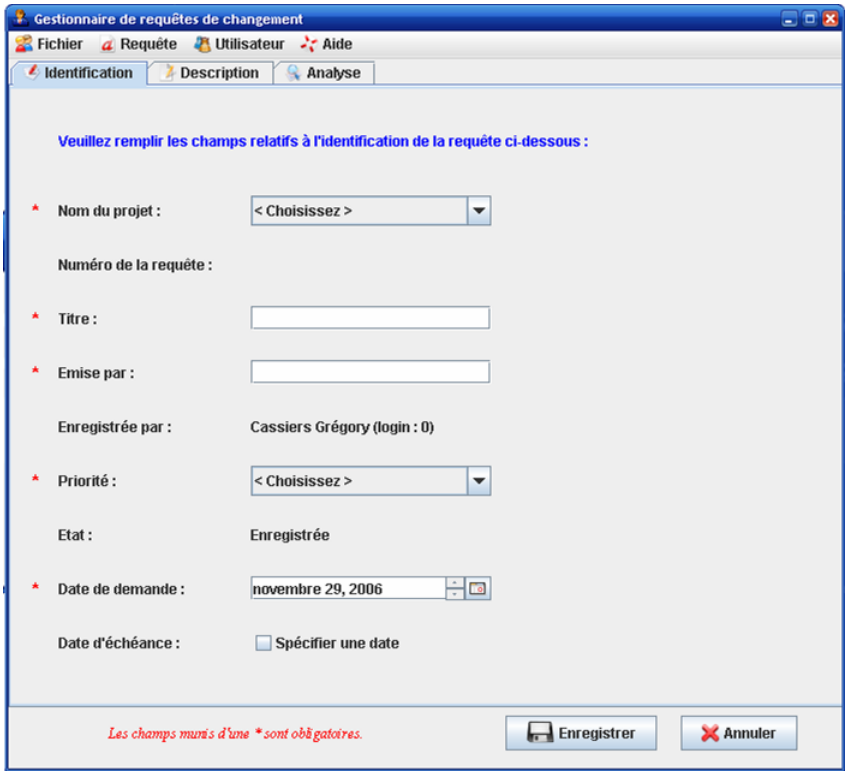
# Menu "Requête"

## Ajouter une requête

L'ajout d'une requête dans le système peut être réalisée par le chef de projet et par l'utilisateur. Cependant, l'utilisateur ne pourra introduire que certaines données relatives à la requête, à savoir, les informations contenues dans la partie Identification et la partie Description.

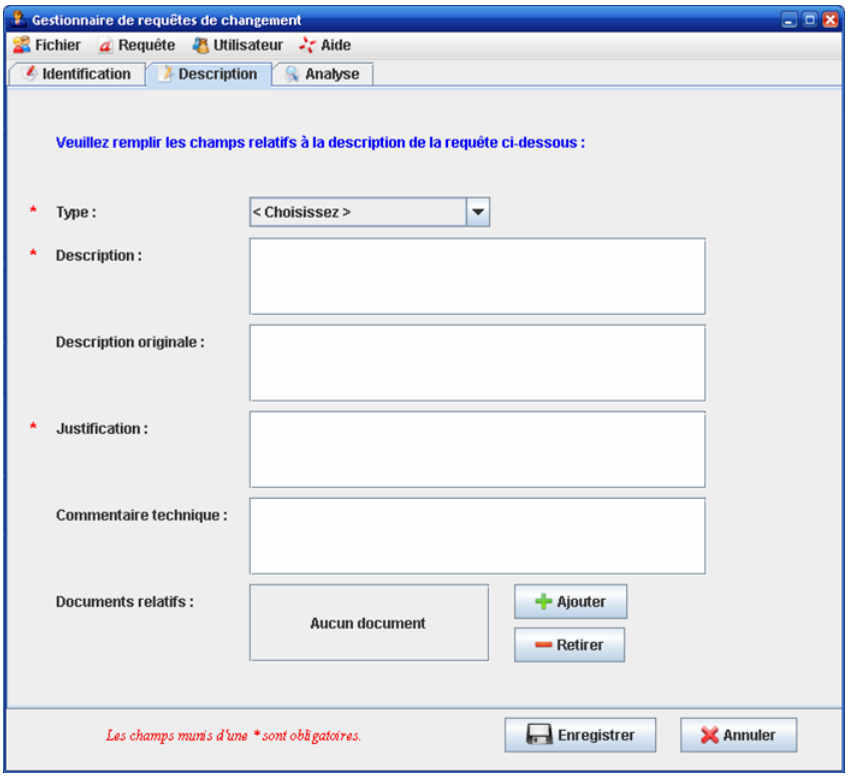
## Identification

Cette partie d'information permet l'identification de la requête. Une requête est identifiée par le projet qu'elle concerne et par son numéro, généré automatiquement et unique dans le projet. Les détails de ces informations sont explicités dans le chapitre 2 de ce document.



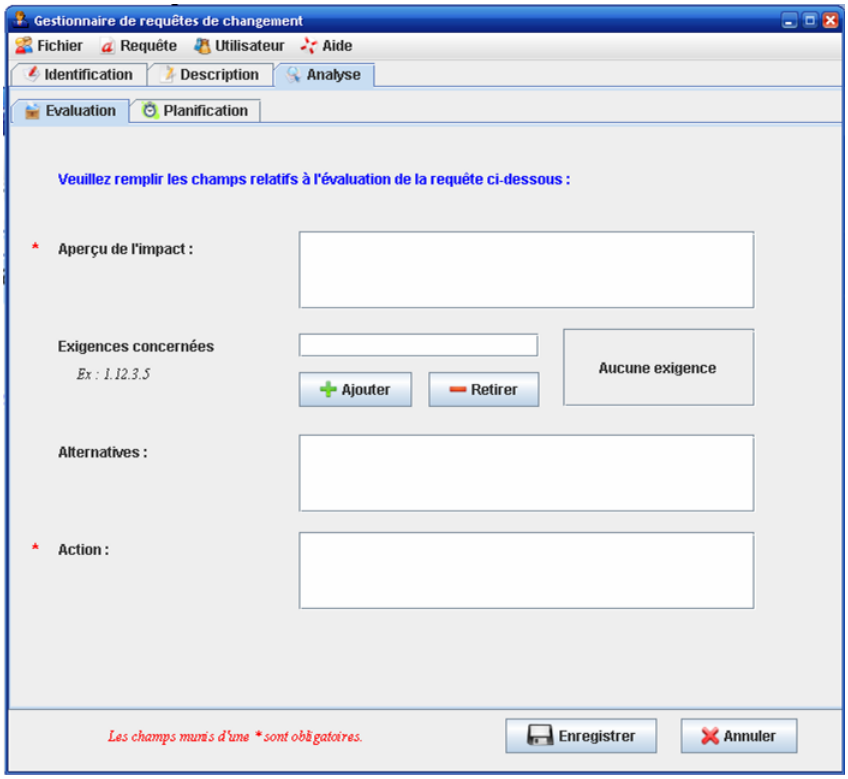
## Description

Cette partie d'information permet la description du changement désiré. Les détails de ces informations sont explicités dans le chapitre 2 de ce document. Remarquons que les noms des documents relatifs peuvent être ajoutés ou retirés aisément via un navigateur de documents en cliquant sur « Ajouter » ou « Retirer ».



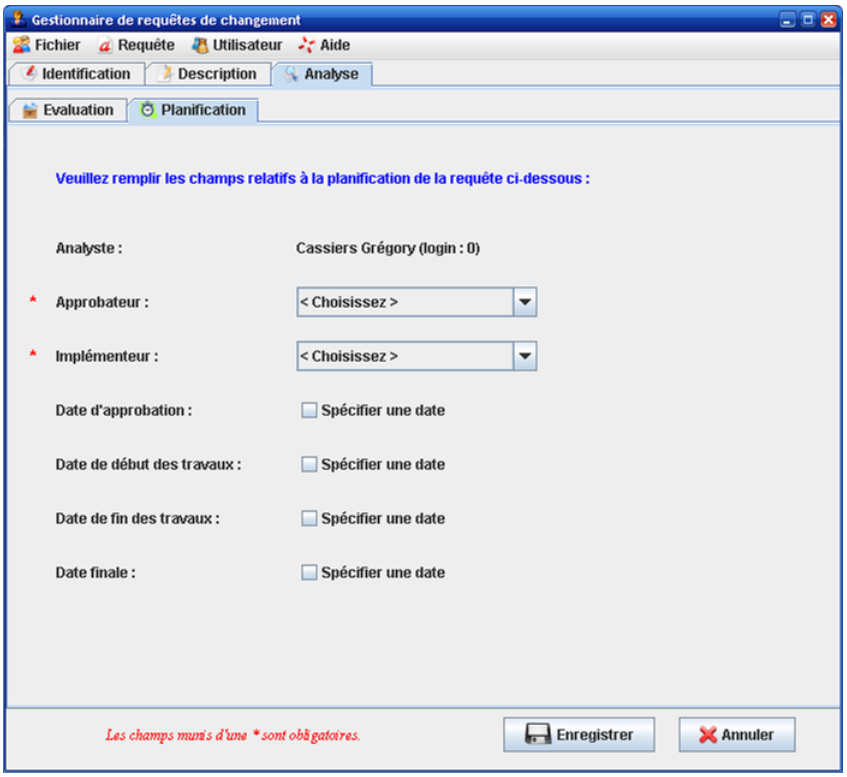
## Analyse - Évaluation

Cette partie d'information permet au chef de projet, et au chef de projet uniquement, de spécifier les informations d'évaluation de la requête. Les détails de ces informations sont explicités dans le chapitre 2 de ce document. Remarquons que les numéros d'exigences peuvent être ajoutés en respectant la syntaxe suivante : x.x.x.x.x.x où x représente un niveau de l'arbre hiérarchique des exigences.



## Analyse - Planification

Cette partie d'information permet au chef de projet, et au chef de projet uniquement, de spécifier les informations de planification du changement. Les détails de ces informations sont explicités dans le chapitre 2 de ce document. Remarquons que l'analyste sera toujours le chef de projet.

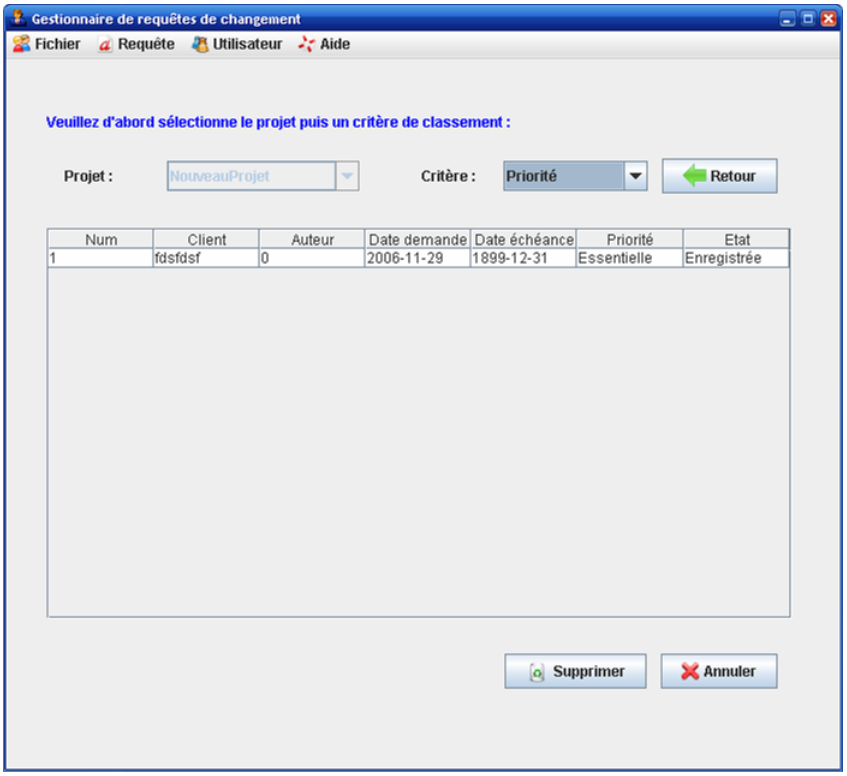


## Modifier une requête

La modification d'une requête dans le système peut être réalisée par le chef de projet et par l'utilisateur. Cependant, comme pour l'ajout d'une requête, l'utilisateur ne pourra modifier que certaines données relatives à la requête, à savoir, les informations contenues dans la partie Identification et la partie Description.

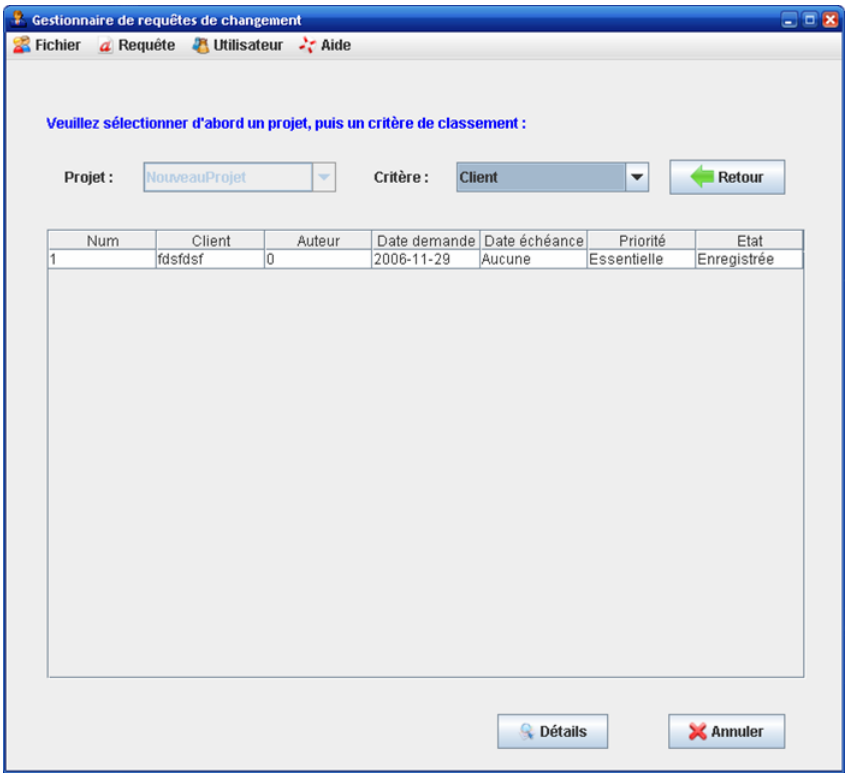
## Supprimer une requête

La suppression d'une requête dans le système peut être réalisée par le chef de projet et par l'utilisateur.



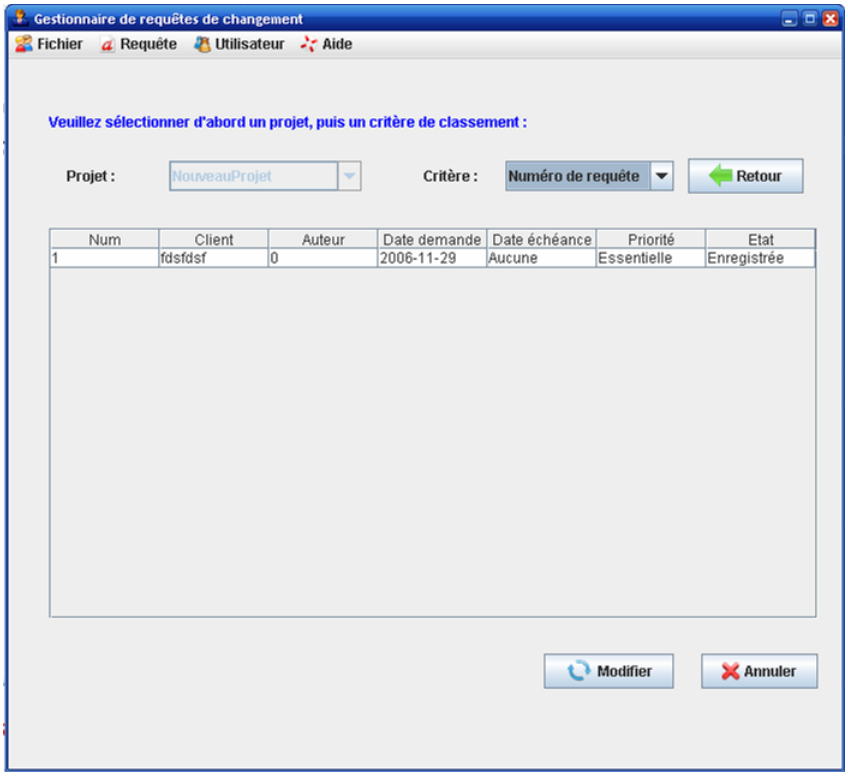
## Consulter les détails d'une requête

Cette fonctionnalité permet au chef de projet et à l'utilisateur de consulter les caractéristiques d'une requête sans pouvoir modifier ces informations. En outre, il est possible de lister les requêtes dépendantes de la requête consultée. Les requêtes dépendantes sont des requêtes qui référencent au moins une exigence référencée par la requête consultée.



## Lister les requêtes

Pour modifier, supprimer ou consulter une requête, l'utilisateur et le chef de projet pourront lister toutes les requêtes d'un projet, et ce, par critère de classement, afin de sélectionner la requête désirée. Le bouton retour permet de sélectionner un autre projet et un autre critère de classement.

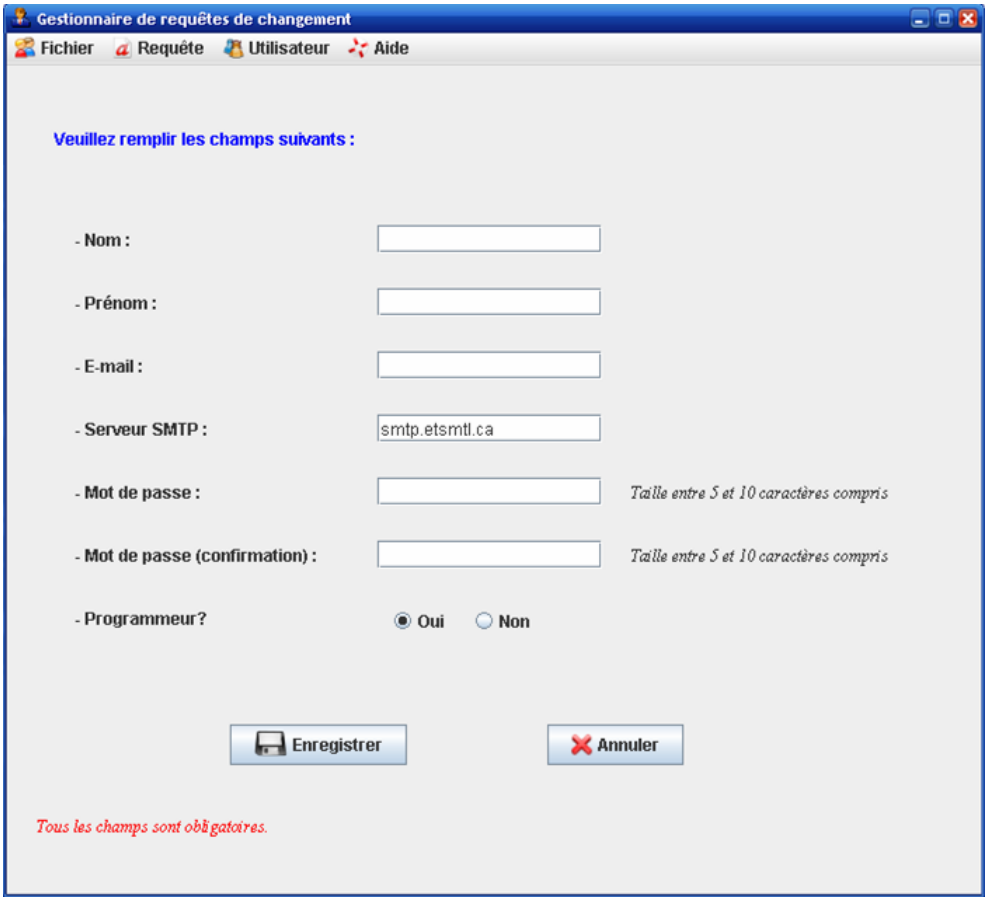


# Menu "Utilisateur"

## Ajouter un utilisateur

Seul le chef de projet a la possibilité d'ajouter des utilisateurs au système.

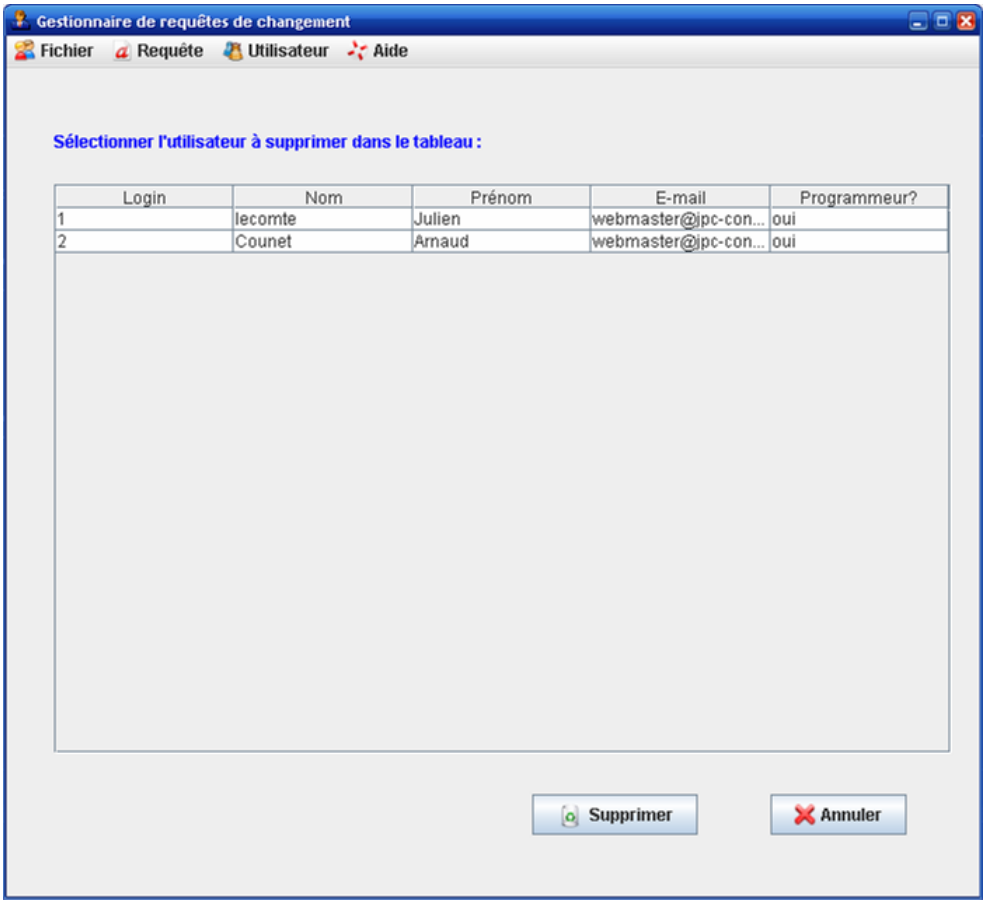
Le chef de projet reçoit les demandes d'inscription au système par mail lorsqu'un utilisateur a, en spécifiant ses informations d'identification, initialisé modeX sur un nouveau poste de travail. Le chef de projet décidera ensuite d'ajouter ou non l'utilisateur concerné en spécifiant les informations de son compte.



## Supprimer un utilisateur

Seul le chef de projet a la possibilité de supprimer des utilisateurs du système.

Lors de la suppression d'un utilisateur par le chef de projet, l'utilisateur est averti par mail de la suppression de son compte.




Si l'utilisateur était impliqué dans une ou plusieurs requêtes de changement, le chef de projet doit remplacer cet utilisateur par un autre, ceci afin d'éviter qu'une requête référence un utilisateur non présent dans le système.